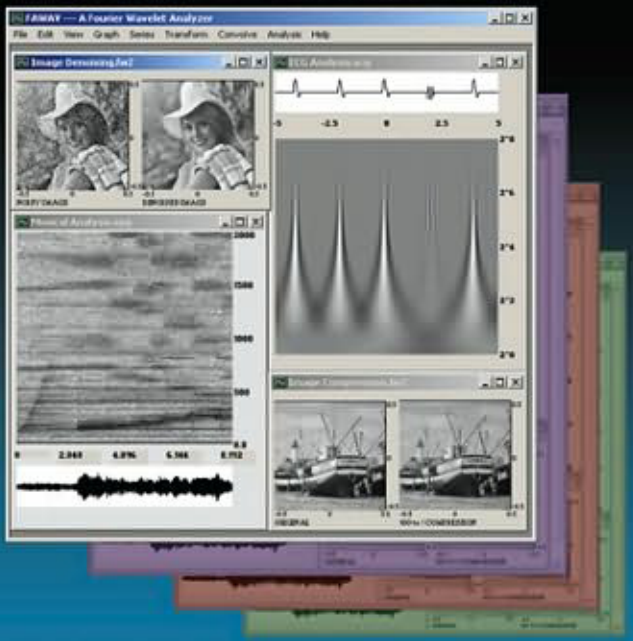
VISIT…

A Primer on WAVELETS and their Scientific Applications

SECOND EDITION



JAMES S. WALKER

Chapman & Hall/CRC
Taylor & Francis Group

A Primer on WAVELETS and Their Scientific Applications

SECOND EDITION

© 2008 by Taylor & Francis Group, LLC

A Primer on WAVELETS and Their Scientific Applications

SECOND EDITION

James S. Walker
University of Wisconsin
Eau Claire, Wisconsin, U.S.A.

Chapman & Hall/CRC
Taylor & Francis Group
Boca Raton London New York

Chapman & Hall/CRC is an imprint of the
Taylor & Francis Group, an **informa** business

© 2008 by Taylor & Francis Group, LLC

Chapman & Hall/CRC
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2008 by Taylor & Francis Group, LLC
Chapman & Hall/CRC is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-58488-745-4 (Softcover)

This book contains information obtained from authentic and highly regarded sources Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The Authors and Publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (http://www.copyright.com/) or contact the Copyright Clearance Center, Inc. (CCC) 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Walker, James S.
A primer on wavelets and their scientific applications / James S. Walker. -- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-58488-745-4 (alk. paper)
1. Wavelets (Mathematics) I. Title.

QA403.3.W33 2008
515'.2433--dc22 2007048858

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

© 2008 by Taylor & Francis Group, LLC

To Berlina and Damian

© 2008 by Taylor & Francis Group, LLC

About the author

James S. Walker received his doctorate from the University of Illinois, Chicago, in 1982. He has been a professor of Mathematics at the University of Wisconsin-Eau Claire since 1982. He has published papers on Fourier analysis, wavelet analysis, complex variables, and logic, and is the author of four books on Fourier analysis, FFTs, and wavelet analysis. He has been married to Angela Huang since 1993, and they are the proud parents of two children, Berlina and Damian, to whom this book is dedicated.

© 2008 by Taylor & Francis Group, LLC

Preface

"Wavelet theory" is the result of a multidisciplinary effort that brought together mathematicians, physicists and engineers...this connection has created a flow of ideas that goes well beyond the construction of new bases or transforms.

—Stéphane Mallat[1]

The past two decades have witnessed an explosion of activity in wavelet analysis. Thousands of research papers have been published on the theory and applications of wavelets. Wavelets provide a powerful and remarkably flexible set of tools for handling fundamental problems in science and engineering. With the second edition of this primer, I aim to provide the most up-to-date introduction to this exciting field.

What is new in this second edition?

I have added a lot of new material to this second edition. The most important additions are the following:

- *Exercises and additional examples:* At the end of the chapters, I provide a total of 104 worked examples, and 222 exercises (with solutions provided for representative problems). These examples and exercises constitute a book-in-itself of review material, and should make this primer a valuable text for courses on wavelets.
- *Biorthogonal wavelets:* I have added two sections that describe the most important examples of these wavelets, and I describe their application to image compression.

[1] Mallat's quote is from [4] listed in the Notes and references section for Chapter 3.

© 2008 by Taylor & Francis Group, LLC

- *Image compression:* I have included a mini-course on image compression, with descriptions of two state-of-the art algorithms (ASWDR and JPEG 2000). This discussion includes a tutorial on arithmetic compression, an important topic that is omitted from most (if not all) introductions to image compression.

- *Image denoising:* I have included a lot more material on image denoising, including a description of a high-performance wavelet denoiser. In addition to the standard topic of removing Gaussian random noise, which most books limit themselves to, I also describe a technique for removing isolated, randomly positioned, clutter from images.

- *Gabor transforms and time-frequency analysis:* I provide a concise introduction to the fundamentals of time-frequency analysis and show how it is applied to musical theory, musical synthesis, and audio denoising. An important new mathematical concept in musical theory, known as the *multiresolution principle,* is described here for the first time in book form. I also provide an extensive list of references to the field of time-frequency analysis, a topic that is given short shrift in most introductions to wavelets.

- *Project ideas:* I provide a list, with brief descriptions, of a set of research projects on wavelets and time-frequency analysis. See Appendix A.

- *Book website:* To keep the book as current as possible, and to provide a wealth of additional online material such as software, sound and image files, and links to other web resources on wavelets, I have created a website for the book. I describe this website in more detail later in the preface.

- *Enhanced list of references:* I have more than doubled the list of references for the book; it now totals about 150 items. At the website for the book I provide an online version of this bibliography with a great many links for free downloading of important papers on wavelets and time-frequency analysis.

What problems can be solved with wavelets?

For an idea of the wide range of problems that can be solved with wavelets, here is a list of some of the problems that I shall discuss:

- *Audio denoising:* Long distance telephone messages, for instance, often contain significant amounts of noise. How do we remove this noise?

- *Signal compression:* The efficient transmission of large amounts of data, over the Internet for example, requires some kind of compression. How do we compress this data without losing significant information?

© 2008 by Taylor & Francis Group, LLC

- *Object detection:* What methods can we use to pick out a small image, say of an aircraft, from the midst of a larger more complicated image?
- *Image denoising:* Images formed by microscopes and other devices are often contaminated by large amounts of unwanted clutter (referred to as *noise*). Can this noise be removed in order to clarify the image?
- *Image enhancement:* When an optical microscope image is recorded, it often suffers from blurring. How can the appearance of the objects in these images be sharpened?
- *Image recognition:* How do humans recognize faces? Can we teach machines to do it?
- *Diagnosing heart trouble:* Is there a way to detect abnormal heartbeats, hidden within a complicated electrocardiogram?
- *Speech recognition:* What factors distinguish consonants from vowels? How do humans recognize different voices?
- *Musical analysis.* How can mathematics help us better understand the nature of musical composition?

All of these problems can be tackled using wavelets. I will show how during the course of this book.

Why do we need another wavelet book?

The goal of this primer is to guide the reader through the main ideas of wavelet analysis in order to facilitate a knowledgeable reading of the present research literature, especially in the applied fields of audio and image processing and biomedicine. Although there are many excellent books on the theory of wavelets, these books are focused on the construction of wavelets and their mathematical properties. Furthermore, they are all written at a graduate school level of mathematical and/or engineering expertise. There remains a real need for a basic introduction, a *primer,* which uses only elementary algebra and a smidgen of calculus to explain the underlying ideas behind wavelet analysis, and devotes the majority of its pages to explaining how these underlying ideas can be applied to solve significant problems in audio and image processing and in biology and medicine.

To keep the mathematics as elementary as possible, I focus on the discrete theory. It is in the continuous theory of wavelet analysis where the most difficult mathematics lies; yet when this continuous theory is applied it is almost always converted into the discrete approach that I describe in this primer. Focusing on the discrete case will allow us to concentrate on the applications of wavelet analysis while at the same time keeping the mathematics under control. On the rare occasions when we need to use more advanced mathematics, *I shall mark these discussions off from the main text by putting them*

© 2008 by Taylor & Francis Group, LLC

into subsections that are marked by asterisks in their titles. An effort has been made to ensure that subsequent discussions do not rely on this more advanced material.

Chapter summaries

Chapter 1 summarizes our subject. Using examples from imaging and audio, I briefly explain wavelet analysis in non-mathematical terms.

In Chapter 2 I introduce the simplest wavelets, the Haar wavelets. I also introduce many of the basic concepts—wavelet transforms, energy conservation and compaction, multiresolution analysis, compression and denoising—that will be used in the remainder of the book. For this reason, I devote more pages to the theory of Haar wavelets than perhaps they deserve alone; keep in mind that this material will be amplified and generalized throughout the remainder of the book.

In Chapter 3 I describe the Daubechies wavelets, which have played a key role in the explosion of activity in wavelet analysis. After a brief introduction to their mathematical properties, I describe several applications of these wavelets. First, I explain in detail how they can be used to compress audio signals—this application is vital to the fields of telephony and telecommunications. Second, I describe how the method of *thresholding* provides a powerful technique for removing random noise (static) from audio signals. Removing random noise is a fundamental necessity when dealing with all kinds of data in science and engineering. The threshold method, which is analogous to how our nervous system responds only to inputs above certain thresholds, provides a nearly optimal method for removing random noise. Besides random noise, Daubechies wavelets can also be used to remove isolated "pop-noise" from audio. The chapter concludes with an introduction to biorthogonal wavelets, which have played an important part in wavelet image compression algorithms such as JPEG 2000.

Wavelet analysis can also be applied to images. In Chapter 4 I provide a mini-course on wavelet-based image compression. I describe two state-of-the-art image compression algorithms, including the new JPEG 2000 standard. I also discuss wavelet-based image denoising, including examples motivated by optical microscopy and scanning tunnelling microscopy. Chapter 4 concludes with some examples from image processing. I examine edge detection, and the sharpening of blurred images, and an example from computer vision where wavelet methods can be used to enormously increase the speed of identification of an image.

Chapter 5 relates wavelet analysis to frequency analysis. Frequency analysis, also known as *Fourier analysis,* has long been one of the cornerstones of the mathematics of science and engineering. I shall briefly describe how wavelets are characterized in terms of their effects on the frequency content of signals. One application that I discuss is object identification—locating a small object within a complicated scene—where wavelet analysis in concert

© 2008 by Taylor & Francis Group, LLC

with Fourier analysis provides a powerful approach. The chapter concludes with a short introduction to the theory and application of Gabor transforms, an important relative of wavelet transforms. In recent years Gabor transforms have been increasingly used in sound processing and denoising, where they may be even more useful than wavelet transforms.

In the final chapter I deal with some extensions which reach beyond the fundamentals of wavelets. I describe a generalization of wavelet transforms known as *wavelet packet transforms.* I apply these wavelet packet transforms to compression of audio signals, images, and fingerprints. Then I turn to the subject of *continuous wavelet transforms*, as they are implemented in a discrete form on a computer. Continuous wavelet transforms are widely used in seismology and have also been used very effectively for analyzing speech and electrocardiograms. We shall also see how they apply to analyzing musical rhythm.

Unfortunately, for reasons of space, I could not examine several important aspects of wavelet theory, such as local cosine series, best-basis algorithms, or fast matrix multiplication. At the end of the final chapter, in its Notes and references section, I provide a guide to the literature for exploring omitted topics.

Computer software

Without question the best way to learn about applications of wavelets is to experiment with making such applications. This experimentation is typically done on a computer. In order to simplify this computer experimentation, I have created free software, called FAWAV. Further details about FAWAV can be found in Appendix C; suffice it to say for now that it is designed to allow the reader to duplicate all of the applications described in this primer and to experiment with other ideas. In this second edition I have added several new features, including image compression and denoising of both images and audio, which should add to the usefulness of FAWAV for digital signal processing.

Notes and references

In the Notes and references sections that conclude each chapter, I provide the reader with ample references where further information can be found. These references are numbered consecutively within each chapter. At the risk of some duplication, but with the reader's convenience in mind, I have also compiled all references into a comprehensive Bibliography.

Primer website

To keep the material in the book as up-to-date as possible, and to facilitate classroom use, there is a website for this book at the following address:

http://www.uwec.edu/walkerjs/Primer/

© 2008 by Taylor & Francis Group, LLC

This website contains a lot of supplementary material for the book, including the following:

- The latest version of FAWAV and its *User Manual.* The *User Manual* is a comprehensive introduction to the use of FAWAV for digital signal processing.
- Papers for downloading, including links to all papers cited in the primer.
- Sound and image files for use with FAWAV, including all the sound and image files referred to in the primer.
- A collection of *Project ideas* for carrying out your own research. This online collection will include an archive of preprints and publications that deal with these projects, and supplement the list from the primer with new topics.
- Links to other websites dealing with wavelet and time-frequency analysis, including all the websites listed in the primer.

Acknowledgments

It is a pleasure to acknowledge several people who have helped me in writing this book. First, I must thank my wife Angela for her constant love and support. I would also like to thank my executive editor, Bob Stern, for his encouragement and help. I appreciate several suggestions from Steve Krantz that helped me put some finishing touches on the book. Thanks to Thomas Maresca for some enlightening discussion on FAWAV and its use in this primer. Thanks also to the UWEC Office of University Research and Creative Activities for a grant that gave me some extra time away from my teaching responsibilities for work on this project. Finally, my students in my Digital Signal Processing and Digital Image Processing courses deserve a warm thank you for all of their many suggestions and questions, which helped me to write a better book. I especially would like to thank Xiaowen Cheng, Jarod Hart, Nicholas Nelson, Derek Olson, and Elizabeth Otteson for their contributions.

James S. Walker
Eau Claire, Wisconsin

© 2008 by Taylor & Francis Group, LLC

Contents

1 Overview **1**
- 1.1 What is wavelet analysis? . . . 1
- 1.2 Notes and references . . . 4

2 Haar wavelets **5**
- 2.1 The Haar transform . . . 6
  - 2.1.1 Haar transform, 1-level . . . 7
- 2.2 Conservation and compaction of energy . . . 9
  - 2.2.1 Conservation of energy . . . 10
  - 2.2.2 Haar transform, multiple levels . . . 11
  - 2.2.3 Justification of conservation of energy . . . 12
- 2.3 Haar wavelets . . . 14
- 2.4 Multiresolution analysis . . . 16
  - 2.4.1 Multiresolution analysis, multiple levels . . . 19
- 2.5 Signal compression . . . 21
  - 2.5.1 A note on quantization . . . 26
- 2.6 Removing noise . . . 26
  - 2.6.1 RMS Error . . . 29
- 2.7 Notes and references . . . 30
- 2.8 Examples and exercises . . . 31

3 Daubechies wavelets **41**
- 3.1 The Daub4 wavelets . . . 41
  - 3.1.1 Remarks on small fluctuation values* . . . 49
- 3.2 Conservation and compaction of energy . . . 50
  - 3.2.1 Justification of conservation of energy* . . . 50
  - 3.2.2 How wavelet and scaling numbers are found* . . . 53
- 3.3 Other Daubechies wavelets . . . 54
  - 3.3.1 Coiflets . . . 58
- 3.4 Compression of audio signals . . . 61
  - 3.4.1 Quantization and the significance map . . . 62
- 3.5 Quantization, entropy, and compression . . . 65
- 3.6 Denoising audio signals . . . 69

© 2008 by Taylor & Francis Group, LLC

3.6.1 Choosing a threshold value . . . 70
3.6.2 Removing pop noise and background static . . . 73
3.7 Biorthogonal wavelets . . . 75
3.7.1 Daub 5/3 system . . . 76
3.7.2 Daub 5/3 inverse . . . 78
3.7.3 MRA for the Daub 5/3 system . . . 78
3.7.4 Daub 5/3 transform, multiple levels . . . 80
3.7.5 Daub 5/3 integer-to-integer system . . . 82
3.8 The Daub 9/7 system . . . 83
3.9 Notes and references . . . 85
3.10 Examples and exercises . . . 87

4 Two-dimensional wavelets 97
4.1 Two-dimensional wavelet transforms . . . 97
4.1.1 Discrete images . . . 98
4.1.2 2D wavelet transforms . . . 99
4.1.3 2D wavelets and scaling images . . . 102
4.2 Compression of images—fundamentals . . . 104
4.2.1 Error measures in image processing . . . 107
4.2.2 Comparing JPEG with wavelet-based compressors . . . 108
4.3 Fingerprint compression . . . 110
4.4 The WDR algorithm . . . 113
4.4.1 Bit-plane encoding . . . 113
4.4.2 Difference reduction . . . 116
4.4.3 Arithmetic compression . . . 119
4.5 The ASWDR algorithm . . . 123
4.5.1 Arithmetic compression . . . 125
4.5.2 Relation to vision . . . 126
4.6 Important image compression features . . . 127
4.6.1 Progressive transmission/reconstruction . . . 127
4.6.2 Lossless compression . . . 128
4.6.3 Region-of-interest . . . 130
4.7 JPEG 2000 image compression . . . 130
4.7.1 Compressing color images . . . 132
4.8 Denoising images . . . 133
4.8.1 The TAWS algorithm . . . 133
4.8.2 Comparison with Wiener denoising . . . 134
4.8.3 Estimation of noise standard deviation * . . . 136
4.8.4 Removal of clutter noise . . . 137
4.9 Some topics in image processing . . . 139
4.9.1 Edge detection . . . 139
4.9.2 Edge enhancement . . . 140
4.9.3 Image recognition . . . 141
4.10 Notes and references . . . 144
4.11 Examples and exercises . . . 147

© 2008 by Taylor & Francis Group, LLC

5 Frequency analysis **167**
5.1 Discrete Fourier analysis . . . 168
5.1.1 Frequency content of wavelets . . . 169
5.2 Definition of the DFT and its properties . . . 170
5.2.1 Properties of the DFT . . . 171
5.2.2 z-transforms * . . . 173
5.3 Frequency description of wavelet analysis . . . 174
5.3.1 Low-pass and high-pass filtering * . . . 178
5.4 Correlation and feature detection . . . 180
5.4.1 DFT method of computing correlations . . . 181
5.4.2 Proof of DFT effect on correlation * . . . 183
5.4.3 Normalized correlations and feature detection * . . . 183
5.5 Object detection in 2D images . . . 185
5.6 Creating scaling signals and wavelets * . . . 188
5.7 Gabor transforms and spectrograms . . . 192
5.8 Musical analysis . . . 195
5.8.1 Analysis of Stravinsky's *Firebird Suite* . . . 197
5.8.2 Analysis of a Chinese folk song . . . 199
5.9 Inverting Gabor transforms . . . 201
5.10 Gabor transforms and denoising . . . 203
5.11 Notes and references . . . 206
5.12 Examples and exercises . . . 210

6 Beyond wavelets **223**
6.1 Wavelet packet transforms . . . 223
6.2 Wavelet packet transforms—applications . . . 225
6.2.1 Fingerprint compression . . . 228
6.3 Continuous wavelet transforms . . . 228
6.4 Gabor wavelets and speech analysis . . . 232
6.4.1 Musical analysis: formants in song lyrics . . . 236
6.5 Percussion scalograms and musical rhythm . . . 237
6.5.1 Analysis of a complex percussive rhythm . . . 241
6.5.2 Multiresolution Principle for rhythm . . . 241
6.6 Notes and references . . . 241
6.6.1 Additional references . . . 242
6.7 Examples and exercises . . . 246

A Projects **255**
A.1 Music . . . 255
A.2 Noise removal from audio . . . 256
A.3 Wavelet image processing . . . 256
A.4 References . . . 257

© 2008 by Taylor & Francis Group, LLC

B Selected exercise solutions **259**
B.1 Introduction 259
B.2 Chapter 2 259
B.3 Chapter 3 262
B.4 Chapter 4 268
B.5 Chapter 5 273
B.6 Chapter 6 279

C Wavelet software **283**
C.1 Installing the book's software 284
C.2 Other software 284
C.3 References 285

Bibliography **287**

© 2008 by Taylor & Francis Group, LLC

Chapter 1

Overview

Thought is impossible without an image.

—Aristotle

We provide a brief, non-mathematical, summary of what wavelet analysis is all about. Examples are discussed from imaging and audio. In the Notes and References section we indicate where some good material on the history of wavelet analysis can be found.

1.1 What is wavelet analysis?

Wavelet analysis decomposes sounds and images into component waves of varying durations, called wavelets. These wavelets are localized vibrations of a sound signal, or localized variations of detail in an image. They can be used for a wide variety of fundamental signal processing tasks, such as compression, removing noise, or enhancing a recorded sound or image in various ways. In this section we give a couple of illustrations of what wavelet analysis involves, chosen from the fields of imaging and audio. In line with the dictum of Aristotle quoted above, we shall first consider the field of imaging.

The heart of wavelet analysis is *multiresolution analysis.* Multiresolution analysis is the decomposition of a signal (such as an image) into subsignals (subimages) of different size resolution levels. As an example, consider how the artist Ray Freer describes his process of painting:

> My approach to watercolor painting is very similar to that with oil or pastel. I start with a broad description of the essential masses and then

© 2008 by Taylor & Francis Group, LLC

> progress through several stages of refinement until I have achieved the final result.[1]

To see what Freer means, look at Figure 1.1. In Figure 1.1(a) we show an initial image, corresponding to the "broad description of the essential masses." Figure 1.1(d) shows the details that need to be added to image (a) in order to obtain the first refinement shown in Figure 1.1(b). These details consist of regions of black where image (a) is to be darkened, and white where image (a) is to be brightened, and a dull gray background where image (a) is left unchanged. If we view these regions as a surface of water (the white parts as higher, the black parts as lower, above the undisturbed gray surface) then the white and black regions in (d) are localized waves, *wavelets,* that represent the changes to be made to image (a). Similarly, in image (e) we show the wavelets representing the changes to be made to image (b) to produce the final image (c). Notice that the wavelets in (e) are finer in detail, smaller scale, than the wavelets in (d).

In wavelet analysis, we begin with the reverse of the process shown in Figure 1.1. We start with the final image and determine the finest scale wavelets contained in it, and remove those to give a less detailed image. We then determine the next finest scale wavelets contained in that image and remove those, and so on, until we arrive at an image consisting of "broad masses," a very low resolution image. This process is called the *wavelet transform* of the image. The wavelet transform thus provides us with the recipe for synthesizing the image as we described above.

What advantages do we gain from this wavelet transform process? To see what we gain in terms of the painting example just discussed, we quote from Enrico Coen's excellent book *The Art of Genes:*

> Why doesn't the artist go directly to the detailed last stage, circumventing the other steps? What purpose do the previous stages serve? One problem with going too quickly into painting details is that they can be easily misplaced. If you start by painting the eyes in great detail, you might find later on that they are not quite in the right place in relation to the other features. You would have to go back and start all over again. By first giving the overall layout and gradually narrowing down to the details, the correct relations between features can be established gradually at each scale, from the overall location of the face, to the position of the eyes, to the details within each eye. To help them achieve this, some artists screw up their eyes when looking at the subject during the early stages of a painting, blurring their vision on purpose so as to more easily see the broad arrangements and avoid being distracted by details.[2]

The last sentence quoted gives us one clue as to how wavelet analysis removes random noise (random clutter) from images. If you "screw up" your eyes to

[1] From [1], page 132.
[2] From [1], page 133.

© 2008 by Taylor & Francis Group, LLC

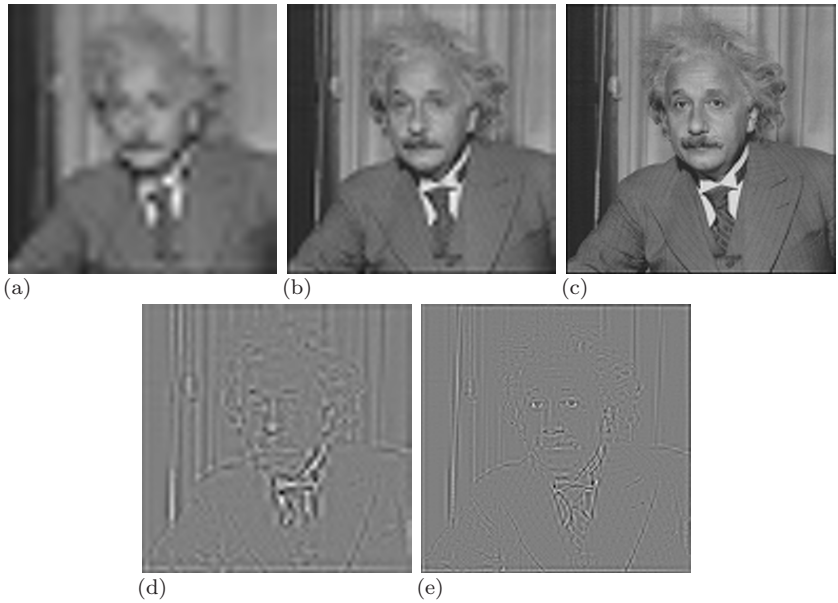


FIGURE 1.1
Drawing a portrait. Figure (a) is the first, rough draft. Figure (d) shows details that are added to (a) to create the second draft (b). In (d), points are blacker if the corresponding points in (a) are to be darkened. They are whiter if the corresponding points in (a) are to be brightened. And they are a dull gray if no change is needed in (a). Similarly, in (e) we show details that are needed to change (b) into the third, and final, draft (c).

blur your vision when looking at one of the noisy images in Chapter 4 [say Figure 4.17(b) on page 137, or Figure 4.18(a) on page 138], you will see that the noise is removed from the blurred image. Of course, there is much more to wavelet-based denoising than that. We will show in Chapter 4 how to add back in details to obtain sharply focused images with the noise removed.

But what does this description of wavelet analysis have to do with compression of images? We can give some idea in the following way. Notice that the wavelets in Figure 1.1(d) are of a significantly larger scale than the ones in Figure 1.1(e). This corresponds to relatively fewer brushstrokes (with a broader brush) needed to refine the image in (a) to produce the image in (b). Likewise, to produce the first draft image (a), even fewer and broader brushstrokes are needed. Because we need progressively fewer brushstrokes at these lower resolutions, the image can be compressed by specifying how to produce the lower resolution, first draft, image and then how to successively refine it (rather than trying to specify how to paint in all the fine details in the original image). We will see in Chapter 4 that this, in essence, is what wavelet transform compression is doing.

While 2D images are easier to see, we will begin our mathematical presentation in the next chapter with 1D signals because the mathematics is simpler

© 2008 by Taylor & Francis Group, LLC

in 1D. The overall process, however, is similar to what we described above for images. The wavelet transform specifies how to remove the finest details from a sound signal (the shortest duration wavelets) to produce a lower resolution signal. But what does "lower resolution" mean for a sound signal? It means a signal where the oscillations are of longer duration in comparison to the short duration wavelets subtracted out from the original. Steven Pinker describes this very well in terms of musical sounds:

> It [musical analysis] dissects the melody into essential parts and ornaments. The ornaments are stripped off and the essential parts further dissected into even more essential parts and ornaments on them... we sense it when we recognize variations of a piece in classical music or jazz. The skeleton of the melody is conserved while the ornaments differ from variation to variation.[3]

The first two sentences of this quotation are an excellent, non-mathematical, summary of what the wavelet transform is doing to 1D signals. Keep it in mind in the next two chapters when we discuss 1D wavelet transforms. We will begin with the Haar wavelet transform in Chapter 2. This transform provides the simplest way to decompose a 1D signal by stripping off details (called *fluctuations*) to obtain a skeleton (called a *trend*). After setting the stage with Haar wavelets, we then turn to the Daubechies wavelet transforms in Chapter 3. These transforms provide a powerful approach to signal compression and signal denoising, and a deep understanding of signal synthesis.

1.2 Notes and references

For those readers interested in the history of wavelet analysis, a good place to start is the article by Burke [3], which has been expanded into a book [4]. There is also some history, of a more sophisticated mathematical nature, in the books by Meyer [5] and Daubechies [6].

1. E. Coen. (1999). *The Art of Genes.* Oxford University Press, Oxford, UK.
2. S. Pinker. (1997). *How the Mind Works.* Norton, NY.
3. B. Burke. (1994). The mathematical microscope: waves, wavelets, and beyond. In *A Positron Named Priscilla, Scientific Discovery at the Frontier,* 196–235, Nat. Academy Press. Edited by M. Bartusiak.
4. B. Burke Hubbard. (1998). *The World According to Wavelets, Second Edition.* AK Peters, Wellesley, MA.
5. Y. Meyer. (1993). *Wavelets. Algorithms and Applications.* SIAM, Philadelphia, PA.
6. I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

[3] From [2], page 533.

© 2008 by Taylor & Francis Group, LLC

Chapter 2

Haar wavelets

The purpose of computing is insight, not numbers.

—Richard W. Hamming

The purpose of computing is insight, not pictures.

—Lloyd N. Trefethen[1]

A Haar wavelet is the simplest type of wavelet. In discrete form, Haar wavelets are related to a mathematical operation called the *Haar transform.* The Haar transform serves as a prototype for all other wavelet transforms. Studying the Haar transform in detail will provide a good foundation for understanding the more sophisticated wavelet transforms which we shall describe in the next chapter. In this chapter we shall describe how the Haar transform can be used for compressing audio signals and for removing noise. Our discussion of these applications will set the stage for the more powerful wavelet transforms to come and their applications to these same problems. One distinctive feature that the Haar transform enjoys is that it lends itself easily to simple hand calculations. We shall illustrate many concepts by both simple hand calculations and more involved computer computations.

[1]Hamming's quote is from [1]. Trefethen's quote is from [2].

© 2008 by Taylor & Francis Group, LLC

2.1 The Haar transform

In this section we shall introduce the basic notions connected with the Haar transform, which we shall examine in more detail in later sections. First, we need to define the type of signals that we shall be analyzing.

We shall be working extensively with *discrete signals.* A discrete signal is a function of time with values occurring at discrete instants. Generally we shall express a discrete signal in the form

$$\mathbf{f} = (f_1, f_2, \ldots, f_N),$$

where N is a positive even integer[2] which we shall refer to as the *length* of $\mathbf{f}$. The *values* of $\mathbf{f}$ are the N real numbers $f_1, f_2, \ldots, f_N$. These values are typically measured values of an analog signal g, measured at the time values $t = t_1, t_2, \ldots, t_N$. That is, the values of $\mathbf{f}$ are

$$f_1 = g(t_1),\ f_2 = g(t_2),\ \ldots,\ f_N = g\left(t_N\right). \tag{2.1}$$

For simplicity, we shall assume that the increment of time that separates each pair of successive time values is always the same. We shall use the phrase *equally spaced sample values*, or just *sample values,* when the discrete signal has its values defined in this way. An important example of sample values is the set of data values stored in a computer audio file, such as a `.wav` file. Another example is the sound intensity values recorded on a compact disc. A non-audio example is a digitized electrocardiogram.

Like all wavelet transforms, the Haar transform decomposes a discrete signal into two subsignals of half its length. One subsignal is a running average or *trend;* the other subsignal is a running difference or *fluctuation.*

Let's begin with the trend. The *first trend*, $\mathbf{a}^1 = (a_1, a_2, \ldots, a_{N/2})$, for the signal $\mathbf{f}$ is computed by taking a running average in the following way. Its first value, a_1, is computed by taking the average of the first pair of values of $\mathbf{f}$: $(f_1+f_2)/2$; and then multiplying it by $\sqrt{2}$. Thus, $a_1 = (f_1+f_2)/\sqrt{2}$. Similarly, its next value a_2 is computed by taking the average of the next pair of values of $\mathbf{f}$: $(f_3 + f_4)/2$; and then multiplying it by $\sqrt{2}$. Hence, $a_2 = (f_3 + f_4)/\sqrt{2}$. Continuing in this way, all of the values of $\mathbf{a}^1$ are produced by taking averages of successive pairs of values of $\mathbf{f}$, and then multiplying these averages by $\sqrt{2}$. A formula for the values of $\mathbf{a}^1$ is

$$a_m = \frac{f_{2m-1} + f_{2m}}{\sqrt{2}}\,, \tag{2.2}$$

for $m = 1, 2, 3, \ldots, N/2$.

For example, suppose $\mathbf{f}$ is defined by eight values, say

$$\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5).$$

[2]If N is odd, then a zero can be appended to the signal to make it even.

© 2008 by Taylor & Francis Group, LLC

The average of its first two values is 5, the average of the next two values is 11, the average of the next two values is 7, and the average of the last two values is 5. Multiplying these averages by $\sqrt{2}$, we obtain the first trend subsignal

$$\mathbf{a}^1 = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2}).$$

You might ask: Why perform the extra step of multiplying by $\sqrt{2}$? Why not just take averages? These questions will be answered in the next section, when we show that multiplication by $\sqrt{2}$ is needed in order to ensure that the Haar transform preserves the energy of a signal.

The other subsignal is called the *first fluctuation.* The first fluctuation of the signal $\mathbf{f}$, which is denoted by $\mathbf{d}^1 = (d_1, d_2, \ldots, d_{N/2})$, is computed by taking a running difference in the following way. Its first value, d_1, is found by taking half the difference of the first pair of values of $\mathbf{f}$: $(f_1 - f_2)/2$; and multiplying it by $\sqrt{2}$. That is, $d_1 = (f_1 - f_2)/\sqrt{2}$. Likewise, its next value d_2 is found by taking half the difference of the next pair of values of $\mathbf{f}$: $(f_3 - f_4)/2$; and multiplying it by $\sqrt{2}$. In other words, $d_2 = (f_3 - f_4)/\sqrt{2}$. Continuing in this way, all of the values of $\mathbf{d}^1$ are produced according to the following formula:

$$d_m = \frac{f_{2m-1} - f_{2m}}{\sqrt{2}}, \tag{2.3}$$

for $m = 1, 2, 3, \ldots, N/2$.

For example, for the signal considered above,

$$\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5),$$

its first fluctuation is

$$\mathbf{d}^1 = (-\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0).$$

2.1.1 Haar transform, 1-level

The Haar transform is performed in several stages, or levels. The first level is the mapping $\mathbf{H}_1$ defined by

$$\mathbf{f} \overset{\mathbf{H}_1}{\longmapsto} (\mathbf{a}^1 \,|\, \mathbf{d}^1) \tag{2.4}$$

from a discrete signal $\mathbf{f}$ to its first trend $\mathbf{a}^1$ and first fluctuation $\mathbf{d}^1$. For example, we showed above that

$$(4, 6, 10, 12, 8, 6, 5, 5) \overset{\mathbf{H}_1}{\longmapsto} (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2} \,|\, -\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0). \tag{2.5}$$

The mapping $\mathbf{H}_1$ in (2.4) has an inverse. Its inverse maps the transform signal $(\mathbf{a}^1 \,|\, \mathbf{d}^1)$ back to the signal $\mathbf{f}$, via the following formula:

$$\mathbf{f} = \left(\frac{a_1 + d_1}{\sqrt{2}}, \frac{a_1 - d_1}{\sqrt{2}}, \ldots, \frac{a_{N/2} + d_{N/2}}{\sqrt{2}}, \frac{a_{N/2} - d_{N/2}}{\sqrt{2}} \right). \tag{2.6}$$

© 2008 by Taylor & Francis Group, LLC

In other words, $f_1 = (a_1 + d_1)/\sqrt{2}$, $f_2 = (a_1 - d_1)/\sqrt{2}$, $f_3 = (a_2 + d_2)/\sqrt{2}$, $f_4 = (a_2 - d_2)/\sqrt{2}$, and so on.

The reader may wish to check that the formulas in (2.6) when applied to

$$(5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2}\,|\,-\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$$

produce

$$(4, 6, 10, 12, 8, 6, 5, 5)$$

thereby inverting the transform in (2.5).

Let's now consider what advantages accrue from performing the Haar transformation. These advantages will be described in more detail later in this chapter, but some basic notions can be introduced now. All of these advantages stem from the following cardinal feature of the Haar transform (a feature that will be even more prominent for the Daubechies transforms described in the next chapter):

Small Fluctuations Feature. *The magnitudes of the values of the fluctuation subsignal are often significantly smaller than the magnitudes of the values of the original signal.*

For instance, for the signal $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$ considered above, its eight values have an average magnitude of 7. On the other hand, for its first fluctuation $\mathbf{d}^1 = (-\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$, the average of its four magnitudes is $0.75\sqrt{2}$. In this case, the magnitudes of the fluctuation's values are an average of 6.6 times smaller than the magnitudes of the original signal's values. For a second example, consider the signal shown in Figure 2.1(a). This signal was generated from 1024 sample values of the function

$$g(x) = 20x^2(1-x)^4 \cos 12\pi x$$

over the interval $[0, 1)$. In Figure 2.1(b) we show a graph of the 1-level Haar transform of this signal. The trend subsignal is graphed on the left half, over the interval $[0, 0.5)$, and the fluctuation subsignal is graphed on the right half, over the interval $[0.5, 1)$. It is clear that a large percentage of the fluctuation's values are close to 0 in magnitude, another instance of the Small Fluctuations Feature. Notice also that the trend subsignal looks like the original signal, although shrunk by half in length and expanded by a factor of $\sqrt{2}$ vertically.

The reason that the Small Fluctuations Feature is generally true is that typically we are dealing with signals sampled from a continuous analog signal g with a very short time increment between the samples. In other words, the equations in (2.1) hold with a small value of the time increment $h = t_{k+1} - t_k$ for each $k = 1, 2, \ldots, N-1$. If the time increment is small enough, then successive values $f_{2m-1} = g(t_{2m-1})$ and $f_{2m} = g(t_{2m})$ of the signal $\mathbf{f}$ will be close to each other due to the continuity of g. Consequently, the fluctuation values for the Haar transform satisfy

$$d_m = \frac{g(t_{2m-1}) - g(t_{2m})}{\sqrt{2}} \approx 0.$$

© 2008 by Taylor & Francis Group, LLC

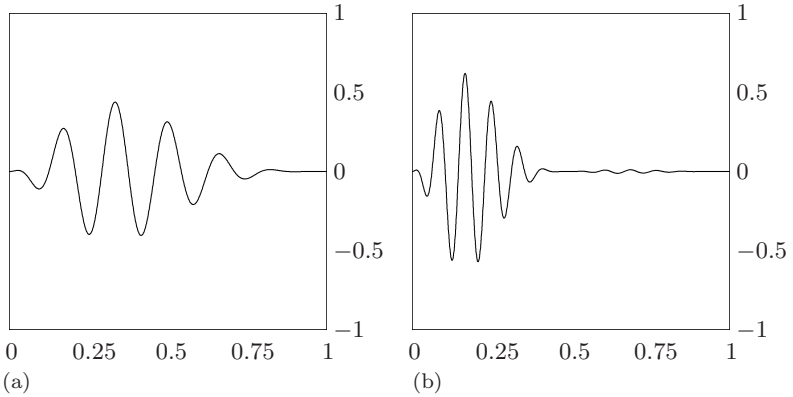


FIGURE 2.1
(a) Signal, (b) Haar transform, 1-level.

This explains why the Small Fluctuations Feature is generally true for the Haar transform. A similar analysis shows why the trend subsignal has a graph that is similar in appearance to the first trend. If g is continuous and the time increment is very small, then $g(t_{2m-1})$ and $g(t_{2m})$ will be close to each other. Expressing this fact as an approximation, $g(t_{2m-1}) \approx g(t_{2m})$, we obtain the following approximation for each value a_m of the trend subsignal

$$a_m \approx \sqrt{2}\, g(t_{2m}).$$

This equation shows that $\mathbf{a}^1$ is approximately the same as sample values of $\sqrt{2}\, g(x)$ for $x = t_2, t_4, \ldots, t_N$. In other words, it shows that the graph of the first trend subsignal is similar in appearance to the graph of g, as we pointed out above in regard to the signal in Figure 2.1(a). We shall examine these points in more detail in the next chapter.

One of the reasons that the Small Fluctuations Feature is important is that it has applications to *signal compression.* By compressing a signal we mean transmitting its values, or approximations of its values, by using a smaller number of bits. For example, if we were only to transmit the trend subsignal for the signal shown in Figure 2.1(a) and then perform Haar transform inversion (treating the fluctuation's values as all zeros), then we would obtain an approximation of the original signal. Since the length of the trend subsignal is half the length of the original signal, this would achieve 50% compression. We shall discuss compression in more detail in Section 2.5.

Once we have performed a 1-level Haar transform, then it is easy to perform multiple-level Haar transforms. We shall discuss this in the next section.

2.2 Conservation and compaction of energy

In the previous section we defined the 1-level Haar transform. In this section we shall discuss its two most important properties: (1) It conserves the ener-

© 2008 by Taylor & Francis Group, LLC

gies of signals; (2) It performs a compaction of the energy of signals. We shall also complete our definition of the Haar transform by showing how to extend its definition to multiple levels.

2.2.1 Conservation of energy

An important property of the Haar transform is that it *conserves the energies of signals.* By the *energy* of a signal $\mathbf{f}$ we mean the sum of the squares of its values. That is, the energy $\mathcal{E}_{\mathbf{f}}$ of a signal $\mathbf{f}$ is defined by

$$\mathcal{E}_{\mathbf{f}} = f_1^2 + f_2^2 + \cdots + f_N^2. \tag{2.7}$$

We shall provide some explanation for why we use the term energy for this quantity $\mathcal{E}_{\mathbf{f}}$ in a moment. First, however, let's look at an example of calculating energy. Suppose $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$ is the signal considered in Section 2.1. Then $\mathcal{E}_{\mathbf{f}}$ is calculated as follows:

$$\mathcal{E}_{\mathbf{f}} = 4^2 + 6^2 + \cdots + 5^2 = 446.$$

Furthermore, using the values for its 1-level Haar transform

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2} \,|\, -\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0),$$

we find that

$$\mathcal{E}_{(\mathbf{a}^1 \,|\, \mathbf{d}^1)} = 25 \cdot 2 + 121 \cdot 2 + \cdots + 2 + 0 = 446$$

as well. Thus the 1-level Haar transform has kept the energy constant. In fact, this is true in general:

Conservation of energy. *The 1-level Haar transform conserves energy. In other words,* $\mathcal{E}_{(\mathbf{a}^1 \,|\, \mathbf{d}^1)} = \mathcal{E}_{\mathbf{f}}$ *for every signal* $\mathbf{f}$.

We will explain why this conservation of energy property is true for all signals at the end of this section.

Before we go any further, we should say something about why we use the term energy for the quantity $\mathcal{E}_{\mathbf{f}}$. The reason is that sums of squares frequently appear in physics when various types of energy are calculated. For instance, if a particle of mass m has a velocity of $\mathbf{v} = (v_1, v_2, v_3)$, then its kinetic energy is $(m/2)(v_1^2 + v_2^2 + v_3^2)$. Hence its kinetic energy is proportional to $v_1^2 + v_2^2 + v_3^2 = \mathcal{E}_{\mathbf{v}}$. Ignoring the constant of proportionality, $m/2$, we obtain the quantity $\mathcal{E}_{\mathbf{v}}$ which we call the energy of $\mathbf{v}$.

While conservation of energy is certainly an important property, it is even more important to consider how the Haar transform redistributes the energy in a signal by compressing most of the energy into the trend subsignal. For example, for the signal $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$ we found in Section 2.1 that its trend $\mathbf{a}^1$ equals $(5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2})$. Therefore, the energy of $\mathbf{a}^1$ is

$$\mathcal{E}_{\mathbf{a}^1} = 25 \cdot 2 + 121 \cdot 2 + 49 \cdot 2 + 25 \cdot 2 = 440.$$

© 2008 by Taylor & Francis Group, LLC

On the other hand, the fluctuation $\mathbf{d}^1$ is $(-\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$, which has energy

$$\mathcal{E}_{\mathbf{d}^1} = 2 + 2 + 2 + 0 = 6.$$

Thus the energy of the trend $\mathbf{a}^1$ accounts for $440/446 = 98.7\%$ of the total energy of the signal. In other words, the 1-level Haar transform has redistributed the energy of $\mathbf{f}$ so that over 98% is concentrated into the subsignal $\mathbf{a}^1$ which is half the length of $\mathbf{f}$. For obvious reasons, this is called *compaction of energy.* As another example, consider the signal $\mathbf{f}$ graphed in Figure 2.1(a) and its 1-level Haar transform shown in Figure 2.1(b). In this case, we find that the energy of the signal $\mathbf{f}$ is 127.308 while the energy of its first trend $\mathbf{a}^1$ is 127.305. Thus 99.998% of the total energy is compacted into the half-length subsignal $\mathbf{a}^1$. By examining the graph in Figure 2.1(b) it is easy to see why such a phenomenal energy compaction has occurred; the values of the fluctuation $\mathbf{d}^1$ are so small, relative to the much larger values of the trend $\mathbf{a}^1$, that its energy $\mathcal{E}_{\mathbf{d}^1}$ contributes only a small fraction of the total energy $\mathcal{E}_{\mathbf{a}^1} + \mathcal{E}_{\mathbf{d}^1}$.

These two examples illustrate the following general principle:

Compaction of energy. *The energy of the trend subsignal* $\mathbf{a}^1$ *accounts for a large percentage of the energy of the transformed signal* $(\mathbf{a}^1 \,|\, \mathbf{d}^1)$.

Compaction of energy will occur whenever the magnitudes of the fluctuation's values are significantly smaller than the trend's values (recall the Small Fluctuations Feature from the last section).

In Section 2.5, we shall describe how compaction of energy provides a framework for applying the Haar transform to compress signals. We now turn to a discussion of how the Haar transform can be extended to multiple levels, thereby increasing the energy compaction of signals.

2.2.2 Haar transform, multiple levels

Once we have performed a 1-level Haar transform, then it is easy to repeat the process and perform multiple level Haar transforms. After performing a 1-level Haar transform of a signal $\mathbf{f}$ we obtain a first trend $\mathbf{a}^1$ and a first fluctuation $\mathbf{d}^1$. The second level of a Haar transform is then performed by computing a second trend $\mathbf{a}^2$ and a second fluctuation $\mathbf{d}^2$ *for the first trend* $\mathbf{a}^1$ *only.*

For example, if $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$ is the signal considered above, then we found that its first trend is $\mathbf{a}^1 = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2})$. To get the second trend $\mathbf{a}^2$ we apply Formula (2.2) *to the values of* $\mathbf{a}^1$. That is, we add successive pairs of values of $\mathbf{a}^1 = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2})$ and divide by $\sqrt{2}$ obtaining $\mathbf{a}^2 = (16, 12)$. To get the second fluctuation $\mathbf{d}^2$, we subtract successive pairs of values of $\mathbf{a}^1 = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2})$ and divide by $\sqrt{2}$ obtaining $\mathbf{a}^2 = (-6, 2)$. Thus the 2-level Haar transform of $\mathbf{f}$ is the signal

$$(\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (16, 12 \,|\, -6, 2 \,|\, -\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0).$$

© 2008 by Taylor & Francis Group, LLC

For this signal $\mathbf{f}$, a 3-level Haar transform can also be done, and the result is

$$(\mathbf{a}^3 \,|\, \mathbf{d}^3 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (14\sqrt{2} \,|\, 2\sqrt{2} \,|\, -6, 2 \,|\, -\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0).$$

It is interesting to calculate the energy compaction that has occurred with the 2-level and 3-level Haar transforms that we just computed. First, we know that $\mathcal{E}_{(\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)} = 446$ because of conservation of energy. Second, we compute that $\mathcal{E}_{\mathbf{a}^2} = 400$. Thus the 2-level Haar transformed signal $(\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)$ has almost 90% of the total energy of $\mathbf{f}$ contained in the second trend $\mathbf{a}^2$ which is 1/4 of the length of $\mathbf{f}$. This is a further compaction, or *localization,* of the energy of $\mathbf{f}$. Furthermore, $\mathcal{E}_{\mathbf{a}^3} = 392$; thus $\mathbf{a}^3$ contains 87.89% of the total energy of $\mathbf{f}$. This is even further compaction; the 3-level Haar transform $(\mathbf{a}^3 \,|\, \mathbf{d}^3 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)$ has almost 88% of the total energy of $\mathbf{f}$ contained in the third trend $\mathbf{a}^3$ which is 1/8 the length of $\mathbf{f}$.

For those readers who are familiar with Quantum Theory, there is an interesting phenomenon here that is worth noting. By Heisenberg's Uncertainty Principle, it is impossible to localize a fixed amount of energy into an arbitrarily small time interval. This provides an explanation for why the energy percentage dropped from 98% to 90% when the second-level Haar transform was computed, and from 90% to 88% when the third-level Haar transform was computed. When we attempt to squeeze the energy into ever smaller time intervals, it is inevitable that some energy leaks out.

As another example of how the Haar transform redistributes and localizes the energy in a signal, consider the graphs shown in Figure 2.2. In Figure 2.2(a) we show a signal, and in Figure 2.2(b) we show the 2-level Haar transform of this signal. In Figures 2.2(c) and (d) we show the respective cumulative energy profiles of these two signals. By the *cumulative energy profile* of a signal $\mathbf{f}$ we mean the signal defined by

$$\left(\frac{f_1^2}{\mathcal{E}_{\mathbf{f}}}, \frac{f_1^2 + f_2^2}{\mathcal{E}_{\mathbf{f}}}, \frac{f_1^2 + f_2^2 + f_3^2}{\mathcal{E}_{\mathbf{f}}}, \ldots, 1 \right).$$

The cumulative energy profile of $\mathbf{f}$ thus provides a summary of the accumulation of energy in the signal as time proceeds. As can be seen from comparing the two profiles in Figure 2.2, the 2-level Haar transform has redistributed and localized the energy of the original signal.

2.2.3 Justification of conservation of energy

We close this section with a brief justification of the conservation of energy property of the Haar transform. First, we observe that the terms a_1^2 and d_1^2

© 2008 by Taylor & Francis Group, LLC

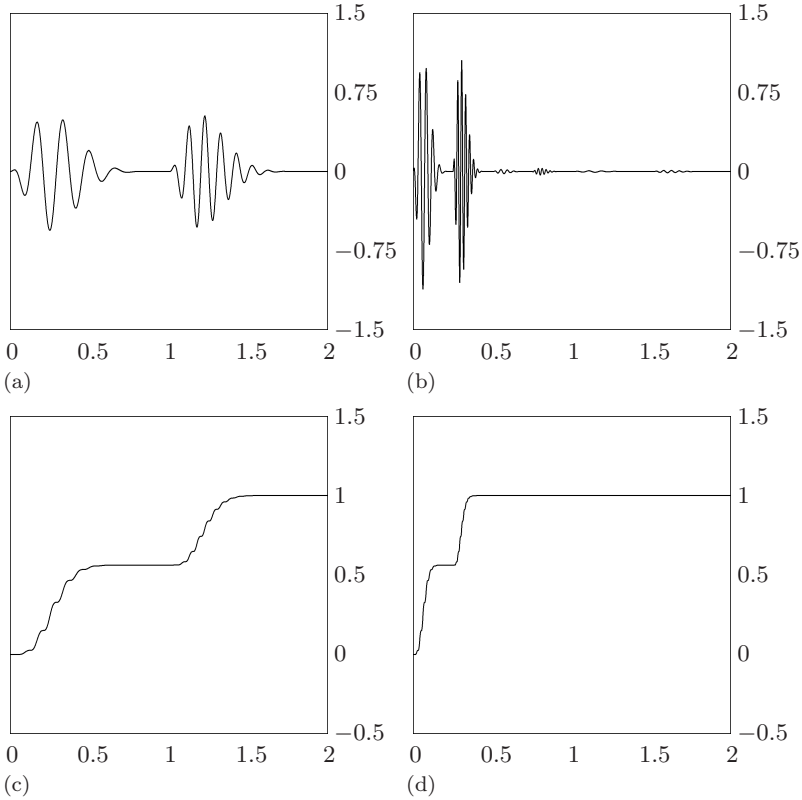


FIGURE 2.2
(a) Signal. (b) 2-level Haar transform of signal. (c) Cumulative energy profile of signal. (d) Cumulative energy profile of 2-level Haar transform.

in the formula $\mathcal{E}_{(\mathbf{a}^1 \mid \mathbf{d}^1)} = a_1^2 + \cdots + a_{N/2}^2 + d_1^2 + \cdots + d_{N/2}^2$ add up as follows:

$$\begin{aligned} a_1^2 + d_1^2 &= \left[\frac{f_1 + f_2}{\sqrt{2}}\right]^2 + \left[\frac{f_1 - f_2}{\sqrt{2}}\right]^2 \\ &= \frac{f_1^2 + 2f_1f_2 + f_2^2}{2} + \frac{f_1^2 - 2f_1f_2 + f_2^2}{2} \\ &= f_1^2 + f_2^2. \end{aligned}$$

Similarly, $a_m^2 + d_m^2 = f_{2m-1}^2 + f_{2m}^2$ for all other values of m. Therefore, by adding a_m^2 and d_m^2 successively for each m, we find that

$$a_1^2 + \cdots + a_{N/2}^2 + d_1^2 + \cdots + d_{N/2}^2 = f_1^2 + \cdots + f_N^2.$$

In other words, $\mathcal{E}_{(\mathbf{a}^1 \mid \mathbf{d}^1)} = \mathcal{E}_{\mathbf{f}}$, which justifies the conservation of energy property.

© 2008 by Taylor & Francis Group, LLC

2.3 Haar wavelets

In this section we discuss the simplest wavelets, the Haar wavelets. This material will set the stage for the more sophisticated Daubechies wavelets described in the next chapter.

We begin by discussing the 1-level *Haar wavelets.* These wavelets are defined as

$$\begin{aligned}
\mathbf{W}_1^1 &= \left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0, \ldots, 0\right) \\
\mathbf{W}_2^1 &= \left(0, 0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0, \ldots, 0\right) \\
\mathbf{W}_3^1 &= \left(0, 0, 0, 0\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0, \ldots, 0\right) \\
&\vdots \\
\mathbf{W}_{N/2}^1 &= \left(0, 0, \ldots, 0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right).
\end{aligned} \tag{2.8}$$

These 1-level Haar wavelets have a number of interesting properties. First, they each have an energy of 1. Second, they each consist of a rapid fluctuation between just two non-zero values, $\pm 1/\sqrt{2}$, with an average value of 0. Hence the name *wavelet.* Finally, they all are very similar to each other in that they are each a translation in time by an even number of time-units of the first Haar wavelet $\mathbf{W}_1^1$. The second Haar wavelet $\mathbf{W}_2^1$ is a translation forward in time by two units of $\mathbf{W}_1^1$, and $\mathbf{W}_3^1$ is a translation forward in time by four units of $\mathbf{W}_1^1$, and so on.

The reason for introducing the 1-level Haar wavelets is that we can express the 1-level fluctuation subsignal in a simpler form by making use of scalar products with these wavelets. The scalar product is a fundamental operation on two signals, and is defined as follows.

Scalar product: *The scalar product* $\mathbf{f} \cdot \mathbf{g}$ *of the signals* $\mathbf{f} = (f_1, f_2, \ldots, f_N)$ *and* $\mathbf{g} = (g_1, g_2, \ldots, g_N)$ *is defined by*

$$\mathbf{f} \cdot \mathbf{g} = f_1 g_1 + f_2 g_2 + \cdots + f_N g_N. \tag{2.9}$$

Using the 1-level Haar wavelets, we can express the values for the first fluctuation subsignal $\mathbf{d}^1$ as scalar products. For example,

$$d_1 = \frac{f_1 - f_2}{\sqrt{2}} = \mathbf{f} \cdot \mathbf{W}_1^1.$$

Similarly, $d_2 = \mathbf{f} \cdot \mathbf{W}_2^1$, and so on. We can summarize Formula (2.3) in terms of scalar products with the 1-level Haar wavelets:

$$d_m = \mathbf{f} \cdot \mathbf{W}_m^1 \tag{2.10}$$

© 2008 by Taylor & Francis Group, LLC

for $m = 1, 2, \ldots, N/2$.

We can also use the idea of scalar products to restate the Small Fluctuations Feature from Section 2.1 in a more precise form. If we say that the *support* of each Haar wavelet is the set of two time-indices where the wavelet is non-zero, then we have the following more precise version of the Small Fluctuations Feature:

Property 1. *If a signal* $\mathbf{f}$ *is (approximately) constant over the support of a* 1*-level Haar wavelet* $\mathbf{W}_k^1$*, then the fluctuation value* $d_k = \mathbf{f} \cdot \mathbf{W}_k^1$ *is (approximately) zero.*

This property will be considerably strengthened in the next chapter.

Note: *From now on, we shall refer to the set of time-indices* m *where* $f_m \neq 0$ *as the* support *of a signal* $\mathbf{f}$*.* For example, the support of the signal $\mathbf{f} = (0, 0, 5, 7, -2, 0, 2, 0)$ consists of the indices $3, 4, 5$, and 7; while the support of the Haar wavelet $\mathbf{W}_2^1$ consists of the indices $3, 4$.

We can also express the 1-level trend values as scalar products with certain elementary signals. These elementary signals are called 1-level *Haar scaling signals,* and they are defined as

$$\begin{aligned}
\mathbf{V}_1^1 &= \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, \ldots, 0\right) \\
\mathbf{V}_2^1 &= \left(0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, \ldots, 0\right) \\
&\vdots \\
\mathbf{V}_{N/2}^1 &= \left(0, 0, \ldots, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right).
\end{aligned} \tag{2.11}$$

Using these Haar scaling signals, the values $a_1, \ldots, a_{N/2}$ for the first trend are expressed as scalar products:

$$a_m = \mathbf{f} \cdot \mathbf{V}_m^1 \tag{2.12}$$

for $m = 1, 2, \ldots, N/2$.

The Haar scaling signals are quite similar to the Haar wavelets. They all have energy 1 and have a support consisting of just two consecutive time-indices. In fact, they are all translates by an even multiple of time-units of the first scaling signal $\mathbf{V}_1^1$. Unlike the Haar wavelets, however, the average values of the Haar scaling signals are not zero. In fact, they each have an average value of $1/\sqrt{2}$.

The ideas discussed above extend to every level. For simplicity, we restrict our discussion to the second level. The 2-level Haar scaling signals are defined

© 2008 by Taylor & Francis Group, LLC

by

$$\begin{aligned}
\mathbf{V}_1^2 &= \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0 \ldots, 0\right) \\
\mathbf{V}_2^2 &= \left(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, \ldots, 0\right) \\
&\vdots \\
\mathbf{V}_{N/4}^2 &= \left(0, 0, \ldots, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right).
\end{aligned} \tag{2.13}$$

These scaling signals are all translations by multiples of four time-units of the first scaling signal $\mathbf{V}_1^2$, and they all have energy 1 and average value 1/2. Furthermore, the values of the 2-level trend $\mathbf{a}^2$ are scalar products of these scaling signals with the signal $\mathbf{f}$. That is, $\mathbf{a}^2$ satisfies

$$\mathbf{a}^2 = \left(\mathbf{f} \cdot \mathbf{V}_1^2, \mathbf{f} \cdot \mathbf{V}_2^2, \ldots, \mathbf{f} \cdot \mathbf{V}_{N/4}^2\right). \tag{2.14}$$

Likewise, the 2-level Haar wavelets are defined by

$$\begin{aligned}
\mathbf{W}_1^2 &= \left(\frac{1}{2}, \frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}, 0, 0 \ldots, 0\right) \\
\mathbf{W}_2^2 &= \left(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}, 0, 0, \ldots, 0\right) \\
&\vdots \\
\mathbf{W}_{N/4}^2 &= \left(0, 0, \ldots, 0, \frac{1}{2}, \frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}\right).
\end{aligned} \tag{2.15}$$

These wavelets all have supports of length 4, since they are all translations by multiples of four time-units of the first wavelet $\mathbf{W}_1^2$. They also all have energy 1 and average value 0. Using scalar products, the 2-level fluctuation $\mathbf{d}^2$ satisfies

$$\mathbf{d}^2 = \left(\mathbf{f} \cdot \mathbf{W}_1^2, \mathbf{f} \cdot \mathbf{W}_2^2, \ldots, \mathbf{f} \cdot \mathbf{W}_{N/4}^2\right). \tag{2.16}$$

2.4 Multiresolution analysis

In the previous section we discussed how the Haar transform can be described using scalar products with scaling signals and wavelets. In this section we discuss how the inverse Haar transform can also be described in terms of these same elementary signals. This discussion will show how discrete signals are synthesized by beginning with a very low resolution signal and successively adding on details to create higher resolution versions, ending with a complete synthesis of the signal at the finest resolution. This is known as *multiresolution analysis* (MRA). MRA is the heart of wavelet analysis.

© 2008 by Taylor & Francis Group, LLC

In order to make these ideas precise, we must first discuss some elementary operations that can be performed on signals. Given two signals $\mathbf{f} = (f_1, f_2, \ldots, f_N)$ and $\mathbf{g} = (g_1, g_2, \ldots, g_N)$, we can perform the following algebraic operations:

Addition and Subtraction: The *sum* $\mathbf{f} + \mathbf{g}$ of the signals $\mathbf{f}$ and $\mathbf{g}$ is defined by adding their values:

$$\mathbf{f} + \mathbf{g} = (f_1 + g_1, f_2 + g_2, \ldots, f_N + g_N). \tag{2.17}$$

Their *difference* $\mathbf{f} - \mathbf{g}$ is defined by subtracting their values:

$$\mathbf{f} - \mathbf{g} = (f_1 - g_1, f_2 - g_2, \ldots, f_N - g_N). \tag{2.18}$$

Constant multiple: A signal $\mathbf{f}$ is multiplied by a constant c by multiplying each of its values by c. That is,

$$c\,\mathbf{f} = (cf_1, cf_2, \ldots, cf_N). \tag{2.19}$$

For example, by repeatedly applying the addition operation, we can express a signal $\mathbf{f} = (f_1, f_2, \ldots, f_N)$ as follows:

$$\mathbf{f} = (f_1, 0, 0, \ldots, 0) + (0, f_2, 0, 0, \ldots, 0) + \cdots + (0, 0, \ldots, 0, f_N).$$

Then, by applying the constant multiple operation to each of the signals on the right side of this last equation, we obtain

$$\mathbf{f} = f_1(1, 0, 0, \ldots, 0) + f_2(0, 1, 0, 0, \ldots, 0) + \cdots + f_N(0, 0, \ldots, 0, 1).$$

This formula is a very natural one; it amounts to expressing $\mathbf{f}$ as a sum of its individual values at each discrete instant of time.

If we define the elementary signals $\mathbf{V}_1^0, \mathbf{V}_2^0, \ldots, \mathbf{V}_N^0$ as

$$\begin{aligned} \mathbf{V}_1^0 &= (1, 0, 0, \ldots, 0) \\ \mathbf{V}_2^0 &= (0, 1, 0, 0, \ldots, 0) \\ &\vdots \\ \mathbf{V}_N^0 &= (0, 0, \ldots, 0, 1) \end{aligned} \tag{2.20}$$

then the last formula for $\mathbf{f}$ can be rewritten as

$$\mathbf{f} = f_1\mathbf{V}_1^0 + f_2\mathbf{V}_2^0 + \cdots + f_N\mathbf{V}_N^0. \tag{2.21}$$

Formula (2.21) is called the *natural expansion* of a signal $\mathbf{f}$ in terms of the *natural basis* of signals $\mathbf{V}_1^0, \mathbf{V}_2^0, \ldots, \mathbf{V}_N^0$. We shall now show that the Haar MRA involves expressing $\mathbf{f}$ as a sum of constant multiples of a different basis set of elementary signals: the Haar wavelets and scaling signals defined in the previous section.

© 2008 by Taylor & Francis Group, LLC

In the previous section, we showed how to express the 1-level Haar transform in terms of wavelets and scaling signals. It is also possible to express the inverse of the 1-level Haar transform in terms of these same elementary signals. This leads to the first level of the Haar MRA. To define this first level Haar MRA we make use of (2.6) to express a signal $\mathbf{f}$ as

$$\mathbf{f} = \left(\frac{a_1}{\sqrt{2}}, \frac{a_1}{\sqrt{2}}, \frac{a_2}{\sqrt{2}}, \frac{a_2}{\sqrt{2}}, \ldots, \frac{a_{N/2}}{\sqrt{2}}, \frac{a_{N/2}}{\sqrt{2}}\right) + \left(\frac{d_1}{\sqrt{2}}, \frac{-d_1}{\sqrt{2}}, \frac{d_2}{\sqrt{2}}, \frac{-d_2}{\sqrt{2}}, \ldots, \frac{d_{N/2}}{\sqrt{2}}, \frac{-d_{N/2}}{\sqrt{2}}\right).$$

This formula shows that the signal $\mathbf{f}$ can be expressed as the sum of two signals that we shall call the first averaged signal and the first detail signal. That is, we have

$$\mathbf{f} = \mathbf{A}^1 + \mathbf{D}^1 \tag{2.22}$$

where the signal $\mathbf{A}^1$ is called the *first averaged signal* and is defined by

$$\mathbf{A}^1 = \left(\frac{a_1}{\sqrt{2}}, \frac{a_1}{\sqrt{2}}, \frac{a_2}{\sqrt{2}}, \frac{a_2}{\sqrt{2}}, \ldots, \frac{a_{N/2}}{\sqrt{2}}, \frac{a_{N/2}}{\sqrt{2}}\right) \tag{2.23}$$

and the signal $\mathbf{D}^1$ is called the *first detail signal* and is defined by

$$\mathbf{D}^1 = \left(\frac{d_1}{\sqrt{2}}, \frac{-d_1}{\sqrt{2}}, \frac{d_2}{\sqrt{2}}, \frac{-d_2}{\sqrt{2}}, \ldots, \frac{d_{N/2}}{\sqrt{2}}, \frac{-d_{N/2}}{\sqrt{2}}\right). \tag{2.24}$$

Using Haar scaling signals and wavelets, and using the basic algebraic operations with signals, the averaged and detail signals are expressible as

$$\mathbf{A}^1 = a_1\mathbf{V}_1^1 + a_2\mathbf{V}_2^1 + \cdots + a_{N/2}\mathbf{V}_{N/2}^1, \tag{2.25a}$$

$$\mathbf{D}^1 = d_1\mathbf{W}_1^1 + d_2\mathbf{W}_2^1 + \cdots + d_{N/2}\mathbf{W}_{N/2}^1. \tag{2.25b}$$

Applying the scalar product formulas for the coefficients in Equations (2.10) and (2.12), we can rewrite these last two formulas as follows

$$\mathbf{A}^1 = (\mathbf{f}\cdot\mathbf{V}_1^1)\mathbf{V}_1^1 + (\mathbf{f}\cdot\mathbf{V}_2^1)\mathbf{V}_2^1 + \cdots + (\mathbf{f}\cdot\mathbf{V}_{N/2}^1)\mathbf{V}_{N/2}^1,$$

$$\mathbf{D}^1 = (\mathbf{f}\cdot\mathbf{W}_1^1)\mathbf{W}_1^1 + (\mathbf{f}\cdot\mathbf{W}_2^1)\mathbf{W}_2^1 + \cdots + (\mathbf{f}\cdot\mathbf{W}_{N/2}^1)\mathbf{W}_{N/2}^1.$$

These formulas show that the averaged signal is a combination of Haar scaling signals, with the values of the first trend subsignal as coefficients; and that the detail signal is a combination of Haar wavelets, with the values of the first fluctuation subsignal as coefficients.

As an example of these ideas, consider the signal

$$\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5).$$

© 2008 by Taylor & Francis Group, LLC

In Section 2.1 we found that its first trend subsignal was

$$\mathbf{a}^1 = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2}).$$

Applying Formula (2.23), the averaged signal is

$$\mathbf{A}^1 = (5, 5, 11, 11, 7, 7, 5, 5). \tag{2.26}$$

Notice how the first averaged signal consists of the repeated average values $5, 5$, and $11, 11$, and $7, 7$, and $5, 5$ about which the values of $\mathbf{f}$ fluctuate. Using Formula (2.25a), the first averaged signal can also be expressed in terms of Haar scaling signals as

$$\mathbf{A}^1 = 5\sqrt{2}\,\mathbf{V}_1^1 + 11\sqrt{2}\,\mathbf{V}_2^1 + 7\sqrt{2}\,\mathbf{V}_3^1 + 5\sqrt{2}\,\mathbf{V}_4^1.$$

Comparing these last two equations we can see that *the positions of the repeated averages correspond precisely with the supports of the scaling signals.*

We also found in Section 2.1 that the first fluctuation signal for $\mathbf{f}$ was $\mathbf{d}^1 = (-\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$. Formula (2.24) then yields

$$\mathbf{D}^1 = (-1, 1, -1, 1, 1, -1, 0, 0).$$

Thus, using the result for $\mathbf{A}^1$ computed above, we have

$$\mathbf{f} = (5, 5, 11, 11, 7, 7, 5, 5) + (-1, 1, -1, 1, 1, -1, 0, 0).$$

This equation illustrates the basic idea of MRA. The signal $\mathbf{f}$ is expressed as a sum of a lower resolution, or averaged, signal $(5, 5, 11, 11, 7, 7, 5, 5)$ added with a signal $(-1, 1, -1, 1, 1, -1, 0, 0)$ made up of fluctuations or details. These fluctuations provide the added details necessary to produce the full resolution signal $\mathbf{f}$.

For this example, using Formula (2.25b), the first detail signal can also be expressed in terms of Haar wavelets as

$$\mathbf{D}^1 = -\sqrt{2}\,\mathbf{W}_1^1 - \sqrt{2}\,\mathbf{W}_2^1 + \sqrt{2}\,\mathbf{W}_3^1 + 0\,\mathbf{W}_4^1.$$

This formula shows that the values of $\mathbf{D}^1$ occur in successive pairs of rapidly fluctuating values positioned at the supports of the Haar wavelets.

2.4.1 Multiresolution analysis, multiple levels

In the discussion above, we described the first level of the Haar MRA of a signal. This idea can be extended to further levels, as many levels as the number of times that the signal length can be divided by 2.

The second level of a MRA of a signal $\mathbf{f}$ involves expressing $\mathbf{f}$ as

$$\mathbf{f} = \mathbf{A}^2 + \mathbf{D}^2 + \mathbf{D}^1. \tag{2.27}$$

© 2008 by Taylor & Francis Group, LLC

Here $\mathbf{A}^2$ is the second averaged signal and $\mathbf{D}^2$ is the second detail signal. Comparing Formulas (2.22) and (2.27) we see that

$$\mathbf{A}^1 = \mathbf{A}^2 + \mathbf{D}^2. \tag{2.28}$$

This formula expresses the fact that computing the second averaged signal $\mathbf{A}^2$ and second detail signal $\mathbf{D}^2$ simply consists of performing a first level MRA of the signal $\mathbf{A}^1$. Because of this, it follows that the second level averaged signal $\mathbf{A}^2$ satisfies

$$\mathbf{A}^2 = (\mathbf{f} \cdot \mathbf{V}_1^2)\mathbf{V}_1^2 + (\mathbf{f} \cdot \mathbf{V}_2^2)\mathbf{V}_2^2 + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N/4}^2)\mathbf{V}_{N/4}^2$$

and the second level detail signal $\mathbf{D}^2$ satisfies

$$\mathbf{D}^2 = (\mathbf{f} \cdot \mathbf{W}_1^2)\mathbf{W}_1^2 + (\mathbf{f} \cdot \mathbf{W}_2^2)\mathbf{W}_2^2 + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N/4}^2)\mathbf{W}_{N/4}^2.$$

For example, if $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$, then we found in Section 2.2 that $\mathbf{a}^2 = (16, 12)$. Therefore

$$\begin{aligned}\mathbf{A}^2 &= 16\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0\right) + 12\left(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) \\ &= (8, 8, 8, 8, 6, 6, 6, 6).\end{aligned} \tag{2.29}$$

It is interesting to compare the equations in (2.26) and (2.29). The second averaged signal $\mathbf{A}^2$ has values created from averages that involve twice as many values as the averages that created $\mathbf{A}^1$. Therefore, the second averaged signal reflects more long term trends than those reflected in the first averaged signal.

We also found in Section 2.2 that this signal $\mathbf{f} = (4, 6, 10, 12, 8, 6, 5, 5)$ has the second fluctuation $\mathbf{d}^2 = (-6, 2)$. Consequently

$$\begin{aligned}\mathbf{D}^2 &= -6\left(\frac{1}{2}, \frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}, 0, 0, 0, 0\right) + 2\left(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{-1}{2}, \frac{-1}{2}\right) \\ &= (-3, -3, 3, 3, 1, 1, -1, -1).\end{aligned}$$

We found above that $\mathbf{D}^1 = (-1, 1, -1, 1, 1, -1, 0, 0)$. Hence

$$\begin{aligned}\mathbf{f} &= \mathbf{A}^2 + \mathbf{D}^2 + \mathbf{D}^1 \\ &= (8, 8, 8, 8, 6, 6, 6, 6) + (-3, -3, 3, 3, 1, 1, -1, -1) \\ &\qquad + (-1, 1, -1, 1, 1, -1, 0, 0).\end{aligned}$$

This formula further illustrates the idea of MRA. The full resolution signal $\mathbf{f}$ is produced from a very low resolution, averaged signal $\mathbf{A}^2$ consisting of repetitions of the two averaged values, 8 and 6, to which are added two detail signals. The first addition supplements this averaged signal with enough details

© 2008 by Taylor & Francis Group, LLC

to produce the next higher resolution averaged signal $(5, 5, 11, 11, 7, 7, 5, 5)$, and the second addition then supplies enough further details to produce the full resolution signal $\mathbf{f}$.

In general, if the number N of signal values is divisible k times by 2, then a k-level MRA:

$$\mathbf{f} = \mathbf{A}^k + \mathbf{D}^k + \cdots + \mathbf{D}^2 + \mathbf{D}^1$$

can be performed on the signal $\mathbf{f}$. Rather than subjecting the reader to the gory details, we conclude by describing a computer example generated using FAWAV. In Figure 2.3 we show a 10-level Haar MRA of the signal $\mathbf{f}$ shown in Figure 2.1(a). This signal has 2^{10} values so 10 levels of MRA are possible. On the top of Figure 2.3(a), the graph of $\mathbf{A}^{10}$ is shown; it consists of a single value repeated 2^{10} times. This value is the average of all 2^{10} values of the signal $\mathbf{f}$. The graph directly below it is of the signal $\mathbf{A}^9$ which equals $\mathbf{A}^{10}$ plus the details in $\mathbf{D}^{10}$. Each successive averaged signal is shown, from $\mathbf{A}^{10}$ through $\mathbf{A}^1$. By successively adding on details, the full signal in Figure 2.1(a) is systematically constructed.

2.5 Signal compression

In Section 2.2 we saw that the Haar transform can be used to localize the energy of a signal into a shorter subsignal. In this section we show how this redistribution of energy can be used to compress audio signals. By compressing an audio signal we mean converting the signal data into a new format that requires less bits to transmit. When we use the term, *audio signal,* we are speaking somewhat loosely. Many of the signals we have in mind are indeed the result of taking discrete samples of a sound signal—as in the data in a computer audio file, or on a compact disc—but the techniques developed here

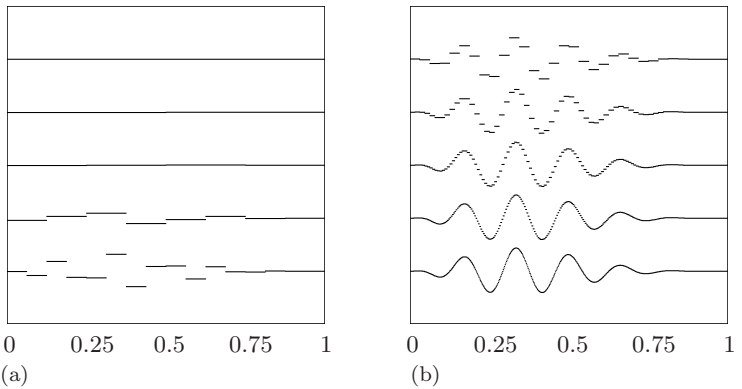


FIGURE 2.3
Haar MRA of the signal in Figure 2.1(a). The graphs are of the ten averaged signals $\mathbf{A}^{10}$ through $\mathbf{A}^1$. Beginning with signal $\mathbf{A}^{10}$ on the top left down to $\mathbf{A}^6$ on the bottom left, then $\mathbf{A}^5$ on the top right down to $\mathbf{A}^1$ on the bottom right.

© 2008 by Taylor & Francis Group, LLC

also apply to digital data transmissions and to other digital signals, such as digitized electrocardiograms or digitized electroencephalograms.

There are two basic categories of compression techniques. The first category is *lossless compression.* Lossless compression methods achieve completely error free decompression of the original signal. Typical lossless methods are Huffman compression, LZW compression, arithmetic compression, or run-length compression. These techniques are used in popular lossless compression programs, such as the kind that produce `.zip` files. Unfortunately, the compression ratios that can be obtained with lossless methods are rarely more than 2:1 for audio files consisting of music or speech.

The second category is *lossy compression.* A lossy compression method is one which produces inaccuracies in the decompressed signal. Lossy techniques are used when these inaccuracies are so small as to be imperceptible. The advantage of lossy techniques over lossless ones is that much higher compression ratios can be attained. With wavelet compression methods, if we are willing to accept slight but imperceptible inaccuracies in the decompressed signal, then we can obtain compression ratios of 10:1, or 20:1, or as high as 50:1 or even 100:1.

In order to illustrate the general principles of wavelet compression of signals, we shall examine, in a somewhat simplified way, how the Haar wavelet transform can be used to compress some test signals. For example, Signal 1 in Figure 2.4(a) can be very effectively compressed using the Haar transform. Although Signal 1 is not a very representative audio signal, it is representative of a portion of a digital data transmission. This signal has 1024 values equally spaced over the time interval $[0, 20)$. Most of these values are constant over long stretches, and that is the principal reason that Signal 1 can be compressed effectively with the Haar transform. Signal 2 in Figure 2.5(a), however, will not compress nearly so well; this signal requires the more sophisticated wavelet transforms described in the next chapter.

The basic steps for wavelet compression are as follows:

Method of Wavelet Transform Compression

Step 1. Perform a wavelet transform of the signal.

Step 2. Set equal to 0 all values of the wavelet transform which are insignificant, i.e., which lie below some *threshold value.*

Step 3. Transmit only the significant, non-zero values of the transform obtained from Step 2. This should be a much smaller data set than the original signal.

Step 4. At the receiving end, perform the inverse wavelet transform of the data transmitted in Step 3, assigning zero values to the insignificant values which were not transmitted. This decompression step produces an approximation of the original signal.

© 2008 by Taylor & Francis Group, LLC

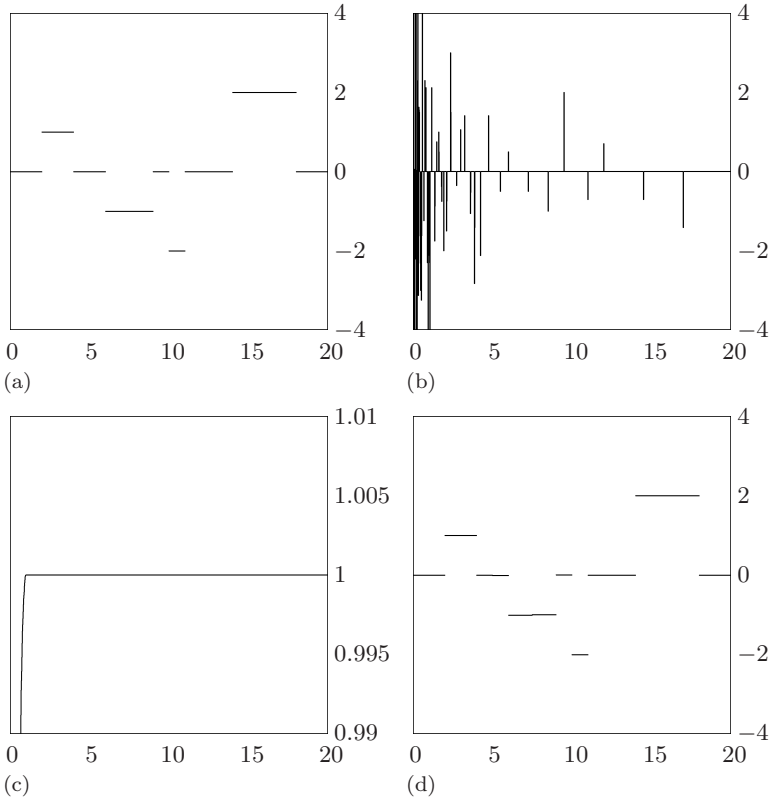


FIGURE 2.4
(a) Signal 1. (b) 10-level Haar transform of Signal 1. (c) Energy map of Haar transform. (d) 20:1 compression of Signal 1, 100% of energy.

In this chapter we shall illustrate this method using the Haar wavelet transform. This initial discussion will be significantly deepened and generalized in the next chapter when we discuss this method of compression in terms of various Daubechies wavelet transforms.

Let's now examine a Haar wavelet transform compression of Signal 1. We begin with Step 1. Since Signal 1 consists of $1024 = 2^{10}$ values, we can perform 10 levels of the Haar transform. This 10-level Haar transform is shown in Figure 2.4(b). Notice how a large portion of the Haar transform's values are 0, or very near 0, in magnitude. This fact provides the fundamental basis for performing an effective compression.

In order to choose a threshold value for Step 2, we proceed as follows. First, we arrange the magnitudes of the values of the Haar transform so that they are in decreasing order:

$$L_1 \geq L_2 \geq L_3 \geq \cdots \geq L_N$$

where L_1 is the largest absolute value of the Haar transform, L_2 is the next

© 2008 by Taylor & Francis Group, LLC

largest, etc. (In the event of a tie, we just leave those magnitudes in their original order.) We then compute the cumulative energy profile of this new signal:

$$\left(\frac{L_1^2}{\mathcal{E}_\mathbf{f}}, \frac{L_1^2+L_2^2}{\mathcal{E}_\mathbf{f}}, \frac{L_1^2+L_2^2+L_3^2}{\mathcal{E}_\mathbf{f}}, \ldots, 1\right).$$

For Signal 1, we show a graph of this energy profile—which we refer to as the *energy map* of the Haar transform—in Figure 2.4(c). Notice that the energy map very quickly reaches its maximum value of 1. In fact, using FAWAV we find that

$$\frac{L_1^2+L_2^2+\cdots+L_{51}^2}{\mathcal{E}_\mathbf{f}} = 0.999996.$$

Consequently, if we choose a threshold T that is less than $L_{51} = 0.3536$, then the values of the transform that survive this threshold will account for essentially 100% of the energy of Signal 1.

We now turn to Step 3. In order to perform Step 3—transmitting only the significant transform values—an additional amount of information must be sent which indicates the positions of these significant transform values in the thresholded transform. This information is called the *significance map*. The values of this significance map are either 1 or 0: a value of 1 if the corresponding transform value survived the thresholding, a value of 0 if it did not. The significance map is therefore a string of N bits, where N is the length of the signal. For the case of Signal 1, with a threshold of 0.35, there are only 51 non-zero bits in the significance map out of a total of 1024 bits. Therefore, since most of this significance map consists of long stretches of zeros, it can be very effectively compressed using one of the lossless compression algorithms mentioned above. This compressed string of bits is then transmitted along with the non-zero values of the thresholded transform.

Finally, we arrive at Step 4. At the receiving end, the significance map is used to insert zeros in their proper locations in between the non-zero values in the thresholded transform, and then an inverse transform is computed to produce an approximation of the signal. For Signal 1 we show the approximation that results from using a threshold of 0.35 in Figure 2.4(d). This approximation used only 51 transform values; so it represents a compression of Signal 1 by a factor of 1024:51, i.e., a compression factor of 20:1. Since the compressed signal contains nearly 100% of the energy of the original signal, it is a very good approximation. In fact, the maximum error over all values is no more than 1.95×10^{-3}.

Life would be simpler if the Haar transform could be used so effectively for all signals. Unfortunately, if we try to use the Haar transform for threshold compression of Signal 2 in Figure 2.5(a), we get poor results. This signal, when played over a computer sound system, produces a sound similar to two low notes played on a clarinet. It has $4096 = 2^{12}$ values; so we can perform 12 levels of the Haar transform. In Figure 2.5(b) we show a plot of the 12-level Haar transform of Signal 2. It is clear from this plot that a large fraction of

© 2008 by Taylor & Francis Group, LLC

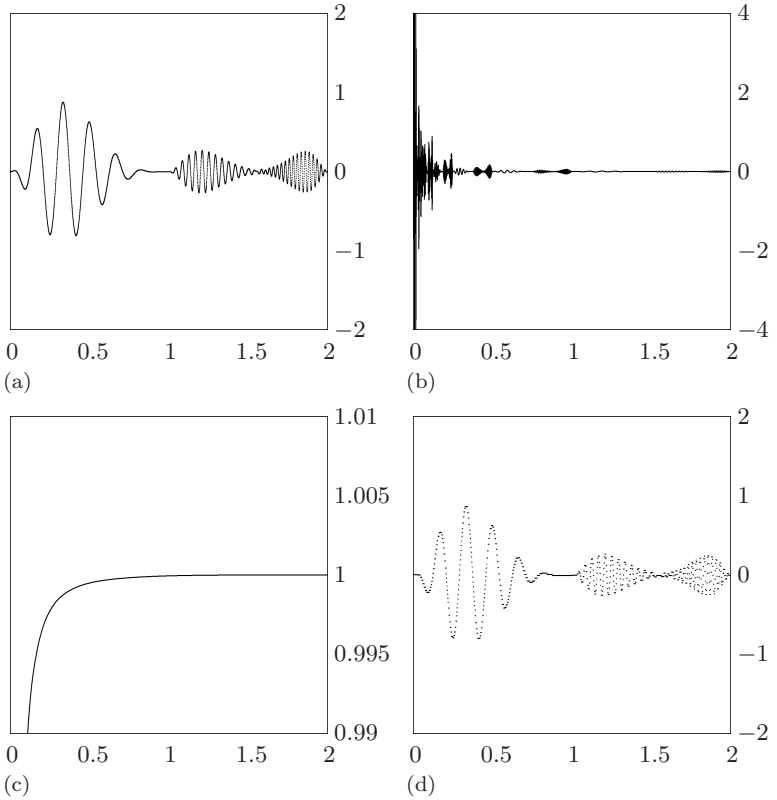


FIGURE 2.5
(a) Signal 2. (b) 12-level Haar transform of Signal 2. (c) Energy map of Haar transform. (d) 10:1 compression of Signal 2, 99.6% of energy.

the Haar transform values have significant magnitude, significant enough that they are visible in the graph. In fact, the energy map for the transform of Signal 2, shown in Figure 2.5(c), exhibits a much slower increase towards 1 in comparison with the energy map for the transform of Signal 1. Therefore, many more transform values are needed in order to capture a high percentage of the energy of Signal 2. In Figure 2.5(d), we show a 10:1 compression of Signal 2 which captures 99.6% of the energy of Signal 2. Comparing this compression with the original signal we see that it is a fairly poor approximation. Many of the signal values are clumped together in the compressed signal, producing a very ragged or jumpy approximation of the original signal. When this compressed version is played on a computer sound system, it produces a screechy "metallic" version of the two clarinet notes, which is not a very satisfying result. As a rule of thumb, we must capture at least 99.99% of the energy of the signal in order to produce an acceptable approximation, i.e., an approximation that is not perceptually different from the original. Achieving this accurate an approximation for Signal 2 requires at least 1782 transform

© 2008 by Taylor & Francis Group, LLC

values. Because Signal 2 itself has 4096 values, this is a compression ratio of only about 2.3:1, which is not very high. We shall see in the next chapter that Signal 2 can be compressed very effectively, but we shall need more high powered wavelet transforms to do it.

2.5.1 A note on quantization

The most serious oversimplification that we made in the discussion above is that we ignored the issue known as *quantization.* The term quantization is used whenever it is necessary to take into account the finite precision of numerical data handled by digital methods. For example, the numerical data used to generate the graphs of Signals 1 and 2 above were IEEE double precision numbers that use 8 bytes = 64 bits for each number. In order to compress this data even further, we can represent the wavelet transform coefficients using less bits. We shall address this issue of quantization in the next chapter when we look again at the problem of compression.

2.6 Removing noise

In this section we shall begin our treatment of one of the most important aspects of signal processing, the removal of noise from signals. Our discussion in this section will introduce the fundamental ideas involved in the context of the Haar transform. In the next chapter we shall considerably deepen and generalize these ideas, in the context of the more powerful Daubechies wavelet transforms.

When a signal is received after transmission over some distance, it is frequently contaminated by noise. The term *noise* refers to any undesired change that has altered the values of the original signal. The simplest model for acquisition of noise by a signal is *additive* noise, which has the form

$$\text{(contaminated signal)} = \text{(original signal)} + \text{(noise)}. \tag{2.30}$$

We shall represent this equation in a more compact way as

$$\mathbf{f} = \mathbf{s} + \mathbf{n} \tag{2.31}$$

where $\mathbf{f}$ is the contaminated signal, $\mathbf{s}$ is the original signal, and $\mathbf{n}$ is the noise.

There are several kinds of noise. A few of the commonly encountered types are the following:

1. *Random noise.* The noise signal is highly oscillatory, its values switching rapidly between values above and below an average, or mean, value. For simplicity, we shall examine random noise with a mean value of 0.

2. *Pop noise.* This type of noise is heard on old analog recordings obtained from phonograph records. The noise is perceived as randomly occurring, isolated "pops." As a model for this type of noise we add a few non-zero values to the original signal at isolated locations.

© 2008 by Taylor & Francis Group, LLC

3. *Localized random noise.* Sometimes the noise appears as in type 1, but only over a short segment or segments of the signal. This occurs when there is a short-lived disturbance in the environment during transmission of the signal.

Of course, there can also be noise signals which combine aspects of each of these types. In this section we shall examine only the first type of noise, random noise. Other types will be considered later.

Our approach will be similar to how we treated compression in the last section; we shall examine how noise removal is performed on two test signals using the Haar transform. For the first test signal, the Haar transform is used very effectively for removing the noise. For the second signal, however, the Haar transform performs poorly, and we shall need to use more sophisticated wavelet transforms to remove the noise from this signal. The essential principles, however, underlying these more sophisticated wavelet methods are the same principles we describe here for the Haar transform.

Here is our basic method for removing random noise.

Threshold Method of Wavelet Denoising

Suppose that the contaminated signal $\mathbf{f}$ equals the transmitted signal $\mathbf{s}$ plus the noise signal $\mathbf{n}$. Also suppose that these two conditions hold:

1. The energy of $\mathbf{s}$ is captured, to a high percentage, by transform values whose magnitudes are all greater than a *threshold* $T_{\mathbf{s}} > 0$.

2. The noise signal's transform values all have magnitudes which lie below a *noise threshold* $T_{\mathbf{n}}$ satisfying $T_{\mathbf{n}} < T_{\mathbf{s}}$.

Then the noise in $\mathbf{f}$ can be removed by thresholding its transform: *All values of its transform whose magnitudes lie below the noise threshold* $T_{\mathbf{n}}$ *are set to* 0 *and an inverse transform is performed, providing a good approximation of* $\mathbf{f}$.

Let's see how this method applies to Signal A shown in Figure 2.6(a). This signal was obtained by adding random noise, whose values oscillate between ± 0.1 with a mean of zero, to Signal 1 shown in Figure 2.4(a). In this case, Signal 1 is the original signal and Signal A is the contaminated signal. As we saw in the last section, the energy of Signal 1 is captured very effectively by the relatively few transform values whose magnitudes lie above a threshold of 0.35. So we set $T_{\mathbf{s}}$ equal to 0.35, and condition 1 in the Denoising Method is satisfied.

Now as for condition 2, look at the 10-level Haar transform of Signal A shown in Figure 2.6(b). Comparing this Haar transform with the Haar transform of Signal 1 in Figure 2.4(b), it is clear that the added noise has contributed a large number of small magnitude values to the transform of Signal A, while the high-energy transform values of Signal 1 are plainly visible

© 2008 by Taylor & Francis Group, LLC

(although slightly altered by the addition of noise). Therefore, we can satisfy condition 2 and eliminate the noise if we choose a noise threshold of, say, $T_{\mathbf{n}} = 0.2$. This is indicated by the two horizontal lines shown in Figure 2.6(b); all transform values lying between ± 0.2 are set to 0, producing the thresholded transform shown in Figure 2.6(c). Comparing Figure 2.6(c) with Figure 2.4(b) we see that the thresholded Haar transform of the contaminated signal is a close match to the Haar transform of the original signal. Consequently, after performing an inverse transform on this thresholded signal, we obtain a denoised signal that is a close match to the original signal. This denoised signal is shown in Figure 2.6(d), and it is clearly a good approximation to Signal 1, especially considering how much noise was originally present in Signal A.

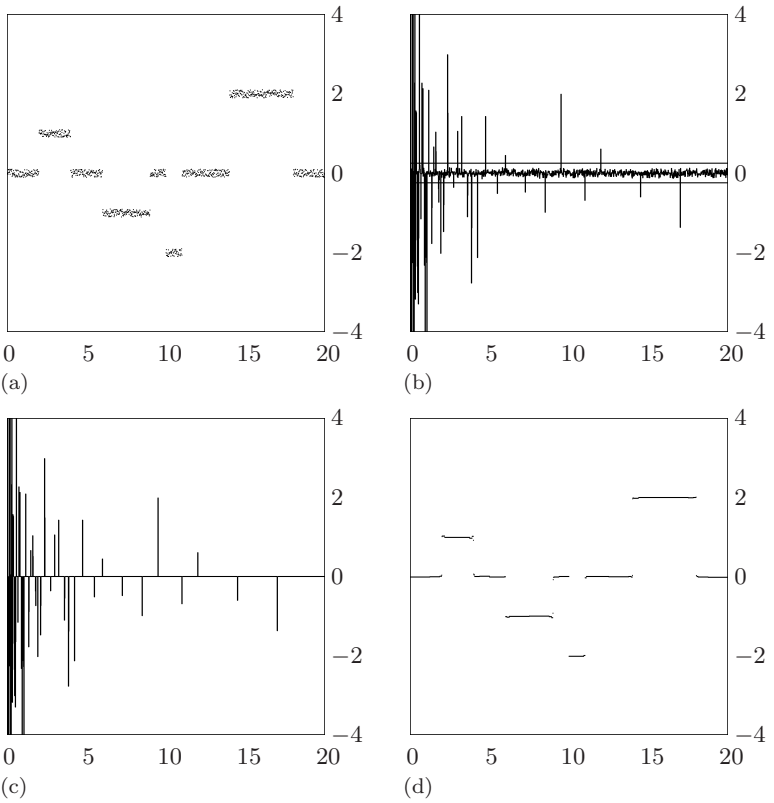


FIGURE 2.6
(a) Signal A, 2^{10} values. (b) 10-level Haar transform of Signal A. The two horizontal lines are at ± 0.2 where 0.2 is a denoising threshold. (c) Thresholded transform. (d) Denoised signal.

© 2008 by Taylor & Francis Group, LLC

2.6.1 RMS Error

The effectiveness of noise removal can be quantitatively measured in the following way. The *Root Mean Square Error* (RMS Error) of the contaminated signal $\mathbf{f}$ compared with the original signal $\mathbf{s}$ is defined to be

$$(\text{RMS Error}) = \sqrt{\frac{(f_1 - s_1)^2 + (f_2 - s_2)^2 + \cdots + (f_N - s_N)^2}{N}}. \qquad (2.32)$$

For example, for Signal A the RMS Error between it and Signal 1 is 0.057. After denoising, the RMS Error between the denoised signal and Signal 1 is 0.011, a five-fold reduction in error. This gives quantitative evidence for the effectiveness of the denoising of Signal A.

Summarizing this example, we can say that the denoising was effective for two reasons: (1) *the transform was able to compress the energy of the original signal into a few high-energy values,* and (2) *the added noise was transformed into low-energy values.* Consequently, the high-energy transform values from the original signal stood out clearly from the low-energy noise transform values which could then be eliminated by thresholding.

Unfortunately, denoising with the Haar transform is not always so effective. Consider, for example, Signal B shown in Figure 2.7(a). This signal consists of Signal 2, shown in Figure 2.5(a), with random noise added. We view Signal 2 as the original signal and Signal B as the contaminated signal. As with the first case considered above, the random noise has values that oscillate between ± 0.1 with a mean of zero. In this case, however, we saw in the last section that it takes a relatively large number of transform values to capture the energy in Signal 2. Most of these transform values are of low energy, and it takes many of them to produce a good approximation of Signal 2. When the random noise is added to Signal 2, then the Haar transform, just like in the previous case, produces many small transform values which lie below a noise threshold. This is illustrated in Figure 2.7(b) where we show the 12-level Haar transform of Signal B. As can be seen by comparing Figure 2.7(b) with Figure 2.5(b), the small transform values that come from the noise *obscure most of the small magnitude transform values that result from the original signal.* Consequently, when a thresholding is done to remove the noise, as indicated by the horizontal lines in Figure 2.7(b), *this removes many of the transform values of the original signal which are needed for an accurate approximation.* This can be verified by comparing the thresholded transform shown in Figure 2.7(c) with the original signal's transform in Figure 2.5(b). In Figure 2.7(d) we show the denoised signal obtained by inverse transforming the thresholded signal. This denoised signal is clearly an unsatisfactory approximation of the original signal. By computing RMS Errors, we can quantify this judgment. The RMS Error between Signal B and Signal 2 is 0.057, while the RMS Error between the denoised signal and Signal 2 is 0.035. This shows that the error after denoising is almost two-thirds as great as the original error.

For this second test case, we can say that the denoising was not effective

© 2008 by Taylor & Francis Group, LLC

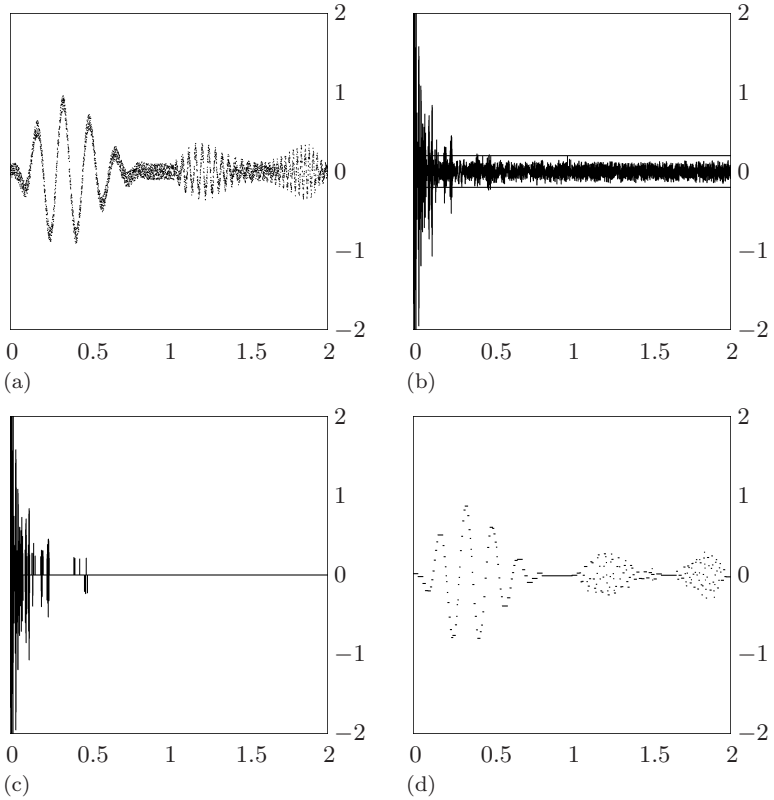


FIGURE 2.7
(a) Signal B, 2^{12} values. (b) 12-level Haar transform of Signal B. The two horizontal lines are at ± 0.2 where 0.2 is the denoising threshold. (c) Thresholded transform. (d) Denoised signal.

because *the transform could not compress the energy of the original signal into a few high-energy values lying above the noise threshold.* We shall see in the next chapter that more sophisticated wavelet transforms can attain the desired compression and thus perform nearly as well at denoising Signal B as the Haar transform did for Signal A.

2.7 Notes and references

More material on the Haar transform and its applications can be found in [3]. Besides the lossy compression method described in this chapter, the Haar transform has also played a role in lossless image compression; see [4] or [5].

1. R.W. Hamming. (1987). *Numerical Methods for Scientists and Engineers.* Dover, New York, NY.
2. L.N. Trefethen. (1998). Maxims about numerical mathematics, computers, science, and life. *SIAM News,* Vol. 31, No. 1.

© 2008 by Taylor & Francis Group, LLC

3. K.R. Rao. (1982). *Fast Transforms: Algorithms, Analyses, Applications.* Academic Press, New York, NY.

4. M. Rabbini and P.W. Jones. (1991). *Digital Image Compression Techniques.* SPIE Press, Bellingham, WA.

5. V.K. Heer and H-E Reinfelder. (1990). Comparison of reversible methods for data compression. In *Medical Imaging IV,* 354–365. Proceedings SPIE, No. 1233.

2.8 Examples and exercises

Note: Computer exercises, designed for FAWAV, are indicated by a superscript **c.** For example, problem 2.1.5 is a computer exercise. A subscript **s** means that a solution is provided. For instance, there is a solution provided for problem 2.1.1(a). Solutions are in Appendix B.

Section 2.1

Example 2.1.A Find the first level Haar transform of $\mathbf{f} = (2, 2, 2, 4, 4, 4)$.

Solution. The average of the first pair of values is 2, the average of the second pair of values is 3, and the average of the third pair of values is 4. Multiplying these averages by $\sqrt{2}$, we obtain $\mathbf{a}^1 = (2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2})$. To compute $\mathbf{d}^1$, we find that $d_1 = (f_1 - f_2)/\sqrt{2} = 0$, and $d_2 = (f_3 - f_4)/\sqrt{2} = -2/\sqrt{2} = -\sqrt{2}$, and $d_3 = (f_5 - f_6)/\sqrt{2} = 0$. Thus the first level Haar transform of $\mathbf{f}$ is $(2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2} \,|\, 0, -\sqrt{2}, 0)$.

Example 2.1.B For the signal $\mathbf{f} = (2, 2, 2, 4, 4, 8)$, compute an approximate signal $\widetilde{\mathbf{f}}$ by inverse transforming the *compressed* Haar transform $(\mathbf{a}^1 \,|\, 0, 0, 0)$ obtained by setting all the fluctuation values equal to zero. Find the largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$.

Solution. We find that $\mathbf{a}^1 = (2\sqrt{2}, 3\sqrt{2}, 6\sqrt{2})$. Therefore, the inverse Haar transform produces $\widetilde{\mathbf{f}} = (2, 2, 3, 3, 6, 6)$. The largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$ is 2.

Example 2.1.C [Figure 2.1] To create Figure 2.1 you do the following. First, choose *New 1-dim* from FAWAV's menu, and then choose *Graph/Plot.* Plot the formula[3]

```
20 x^2 (1-x)^4 cos(12 pi x)
```

over the interval of type $[0, L]$ with $L = 1$ using 1024 points. That produces the graph in Figure 2.1(a) (after selecting *View/Display style* and choosing *Blank* for the Grid style and *Lines* for the Plot style). To produce Figure 2.1(b), select *Transforms/Wavelet* and choose *Haar* as the Wavelet type with 1 entered for the Levels value. After plotting the transform, change to *Lines* as the plot style to get the graph shown in the figure.

2.1.1 Compute the first trend and first fluctuation for the following signals:

(a)$_{\mathbf{s}}$ $\mathbf{f} = (2, 4, 6, 6, 4, 2)$

(b) $\mathbf{f} = (-1, 1, 2, -2, 4, -4, 2, 2)$

[3]This formula is saved in the archive `Primer_Formulas.zip` which can be downloaded from the book's website. After extracting the formulas from this archive, you can retrieve them using the *Load* button under the text box in the graphing procedure.

© 2008 by Taylor & Francis Group, LLC

(c)$_{\mathbf{s}}$ $\mathbf{f} = (1, 2, 3, 3, 2, 2, 1, 1)$

(d) $\mathbf{f} = (2, 2, 4, 4, 6, 6, 8, 8)$

2.1.2 Given the following Haar transformed signals, find the original signals $\mathbf{f}$ that correspond to them using Formula (2.6).

(a)$_{\mathbf{s}}$ $(2\sqrt{2}, -\sqrt{2}, 3\sqrt{2}, -\sqrt{2} \,|\, 0, \sqrt{2}, 0, \sqrt{2})$

(b) $(4\sqrt{2}, 3\sqrt{2}, -\sqrt{2}, 2\sqrt{2} \,|\, \sqrt{2}, -\sqrt{2}, 0, 2\sqrt{2})$

(c)$_{\mathbf{s}}$ $(3\sqrt{2}, 2\sqrt{2}, 2\sqrt{2}, 0 \,|\, 2\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$

(d) $(4\sqrt{2}, 5\sqrt{2}, 7\sqrt{2}, -4\sqrt{2} \,|\, \sqrt{2}, 2\sqrt{2}, -2\sqrt{2}, \sqrt{2})$

2.1.3 For each of the signals $\mathbf{f}$ given below, compute an approximate signal $\widetilde{\mathbf{f}}$ by inverse transforming the *compressed* Haar transform $(\mathbf{a}^1 \,|\, 0, \ldots, 0)$ obtained by setting all the fluctuation values equal to zero. In each case, find the largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$.

(a)$_{\mathbf{s}}$ $\mathbf{f} = (2, 2, 3, 3, 4, 5, 6, 6)$

(b) $\mathbf{f} = (1, 2, 3, 3, 2, 1)$

(c)$_{\mathbf{s}}$ $\mathbf{f} = (2, -2, -2, -2, -2, 0, 2, 2)$

(d) $\mathbf{f} = (4, 4, 4, -1, -1, 1, 2, 2, 4, 6)$

2.1.4 Consider again problem 2.1.3 above. When will there be a difference between a value of $\mathbf{f}$ and a value of the approximate signal $\widetilde{\mathbf{f}}$, and when will the two signals' values be the same?

2.1.5$^{\mathbf{c}}$ Plot 1-level Haar transforms of the following functions—sampled uniformly over $[0, 1]$ using 1024 points.

(a) $f(x) = x^2(1 - x)$

(b)$_{\mathbf{s}}$ $f(x) = x^4(1 - x)^6 \cos(64\pi x)$

(c) $(0.2 < x < 0.3) - 3(0.4 < x < 0.5) + 2(0.5 < x < 0.8)$

(d) $f(x) = \operatorname{sgn}(\sin 12\pi x)$

Section 2.2

Example 2.2.A For the signal $\mathbf{f} = (2, 2, 4, 6, 8, 10)$, find the energies of its trend and fluctuation subsignals and show that their sum equals the energy of $\mathbf{f}$.

Solution. The trend is $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, 9\sqrt{2})$ and fluctuation is $\mathbf{d}^1 = (0, -\sqrt{2}, -\sqrt{2})$. The trend energy $\mathcal{E}_{\mathbf{a}^1}$ is $8 + 50 + 162 = 220$, and the fluctuation energy is $\mathcal{E}_{\mathbf{d}^1} = 0+2+2 = 4$. Their sum is 224 and the energy of $\mathbf{f}$ is $4+4+16+36+64+100 = 224$ so they are equal.

Example 2.2.B Compute the percentage of compaction of the energy of $\mathbf{f} = (2, 2, 4, 6, 8, 10)$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_{\mathbf{f}}$.

Solution. In Example 2.2.A, we found that $\mathcal{E}_{\mathbf{a}^1} = 220$ and that $\mathcal{E}_{\mathbf{f}} = 224$. Therefore, $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_{\mathbf{f}} = 220/224 = 0.982\ldots$

© 2008 by Taylor & Francis Group, LLC

Example 2.2.C [Figure 2.2] To graph Figure 2.2(a), you plot (after selecting *Edit/Points used* and selecting 4096 as the number of points):

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

over the interval $[0, 2]$, using *Lines* as the Plot style. Figure 2.2(b) is plotted by performing a Haar wavelet transform with 2 for the number of Levels (after changing the Grid style to *Blank* and the Y-interval values to $-1.5, 1.5$). Figure 2.2(c) is generated by right-clicking on the graph for the function [Figure 2.2(a)] and selecting *Energy graph*; the resulting graph is then clipped out by right-clicking and selecting *Clip* and entering 2 for the graph to be clipped. Similarly, Figure 2.2(d) is created by performing the same steps for the Haar transform graph.

Example 2.2.D For the signal $\mathbf{f} = (2, 2, 4, 6, 8, 8, 12, 10)$, find its 1-level, 2-level, and 3-level Haar transforms.

Solution. The pairwise, successive averages for $\mathbf{f}$ are 2, 5, 8, 11. Hence $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, 8\sqrt{2}, 11\sqrt{2})$. In a similar way, we find that $\mathbf{d}^1 = (0, -\sqrt{2}, 0, \sqrt{2})$. Thus, the 1-level Haar transform of $\mathbf{f}$ is

$$(\mathbf{a}^1 \mid \mathbf{d}^1) = (2\sqrt{2}, 5\sqrt{2}, 8\sqrt{2}, 11\sqrt{2} \mid 0, -\sqrt{2}, 0, \sqrt{2}).$$

By applying the 1-level Haar transform to $\mathbf{a}^1$, we obtain

$$\mathbf{a}^1 \stackrel{\mathbf{H}_1}{\longmapsto} (\mathbf{a}^2 \mid \mathbf{d}^2) = (7, 19 \mid -3, -3).$$

Hence the 2-level Haar transform of $\mathbf{f}$ is

$$\mathbf{f} \stackrel{\mathbf{H}_2}{\longmapsto} (7, 19 \mid -3, -3 \mid 0, -\sqrt{2}, 0, \sqrt{2}).$$

Applying the 1-level Haar transform to $\mathbf{a}^2$ we obtain $(7, 19) \stackrel{\mathbf{H}_1}{\longmapsto} (13\sqrt{2} \mid -6\sqrt{2})$. Thus, the 3-level transform is

$$\mathbf{f} \stackrel{\mathbf{H}_3}{\longmapsto} (13\sqrt{2} \mid -6\sqrt{2} \mid -3, -3 \mid 0, -\sqrt{2}, 0, \sqrt{2}).$$

2.2.1$_\mathbf{s}$ Find the energies of the trend and fluctuation subsignals for the signals given in problems 2.1.1 (a)–(d), and show that their sums are equal to the energies of the signals $\mathbf{f}$.

2.2.2 Compute the percentage of compaction of the energy of $\mathbf{f}$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f}$, for each of the signals $\mathbf{f}$ in problem 2.1.1(a)–(d).

2.2.3$_\mathbf{s}$ Another way to measure compaction is by the percentage of Haar transform values that are less than some small, preassigned number ϵ (> 0). Compute the percentage of 1-level Haar transform values which are less than $\epsilon = 0.05$ for each of the signals $\mathbf{f}$ in problem 2.1.1(a)–(d).

2.2.4 For problem 2.2.3, change the value of ϵ to $\epsilon = 0.05\,\mathcal{E}_\mathbf{f}$ (so that ϵ depends on $\mathcal{E}_\mathbf{f}$) and recompute the percentage of compaction of the energy of $\mathbf{f}$ by the 1-level Haar transform for each of the signals $\mathbf{f}$ in problem 2.1.1 (a)–(d).

© 2008 by Taylor & Francis Group, LLC

2.2.5$^{c}_{s}$ Find the energies of the trend and fluctuation subsignals for the signals given in problem 2.1.5 (a)–(d), and show that their sums are equal to the energies of the signals **f**. [Hint: In FAWAV, use *Analysis/Statistics* to compute energies for discrete signals. To compute $\mathcal{E}_{\mathbf{a}^1}$ you can modify the 1-level transform by plotting the function `g1(x)(x<1/2)` and then calculating the energy of the resulting signal. Similarly, the function `g1(x)(x >= 1/2)` will plot the first fluctuation.]

2.2.6^{c} Using $\epsilon = 0.05\,\mathcal{E}_{\mathbf{f}}$ compute the percentage of compaction of the energy of **f** by the 1-level Haar transform, in terms of the percentage of transform values with magnitude less than ϵ, for each of the signals **f** in problem 2.1.5 (a)–(d). [*Hint:* You can also plot a graph indicating where x-values are smaller in magnitude than some number c by using the formula `(abs(g1(x))<c)`.]

2.2.7 Find the 1-level, 2-level, and 3-level Haar transforms of the following signals.

(a) $(8, 32, 48, 48, 64, 64, 40, -8)$

(b)$_{\mathbf{s}}$ $(-16, -16, -16, 32, 48, 48, 96, 96)$

(c) $(16, 16, 32, 48, 64, 72, 72, 72, 72, -16, -64, -32, -32, 40, 64, 16)$

(d) $(8, 8, 0, -8, -16, 24, 24, 24, 24, 16, 8, 0, 8, 8, 8, -16)$

Section 2.3

Example 2.3.A For $\mathbf{f} = (3, 2, -1, 4)$ and $\mathbf{g} = (2, -2, 2, -2)$, find the scalar product $\mathbf{f} \cdot \mathbf{g}$.

Solution. $\mathbf{f} \cdot \mathbf{g} = 3(2) + 2(-2) - 1(2) + 4(-2) = -8$.

Example 2.3.B Compute the inverse 1-level Haar transform of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0),$$

and of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 0, \ldots, 0 \,|\, 0, 1, 0, \ldots, 0).$$

Solution. For the inverse transform of $(0, 1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0)$, we find that $f_3 = (a_2 + d_2)/\sqrt{2} = 1/\sqrt{2}$ and $f_4 = (a_2 - d_2)/\sqrt{2} = 1/\sqrt{2}$ and all other values of **f** are 0. Thus the inverse transform is $(0, 0, \sqrt{2}/2, \sqrt{2}/2, 0, 0, \ldots, 0)$.

For the inverse transform of $(0, 0, \ldots, 0 \,|\, 0, 1, 0, \ldots, 0)$ we find that $f_3 = (a_2 + d_2)/\sqrt{2} = 1/\sqrt{2}$ and $f_4 = (a_2 - d_2)/\sqrt{2} = -1/\sqrt{2}$ and all other values of **f** are 0. Thus the inverse transform is $(0, 0, \sqrt{2}/2, -\sqrt{2}/2, 0, 0, \ldots, 0)$.

2.3.1 Find the scalar product $\mathbf{f} \cdot \mathbf{g}$ when **f** and **g** are the following:

(a)$_{\mathbf{s}}$ $\mathbf{f} = (2, 1, 3, 4)$, $\mathbf{g} = (-1, 2, -2, 1)$

(b) $\mathbf{f} = (3, 2, -1, 2)$, $\mathbf{g} = (1, 3, 1, -1)$

(c)$_{\mathbf{s}}$ $\mathbf{f} = (1, 1, -1, -1, 2, 2, -2, 2)$, $\mathbf{g} = (0, 1, 0, 1, 2, 2, 1, 1)$

(d) $\mathbf{f} = (1, 1, 3)$, $\mathbf{g} = (1, 0, 2)$

© 2008 by Taylor & Francis Group, LLC

2.3.2 Prove Property 1.

2.3.3 When will a second fluctuation value d_k^2 equal zero?

2.3.4 Is the following statement true or false? *Whenever the fluctuation value* $d_1^2 = 0$*, then the signal* $\mathbf{f}$ *is constant over the support of* $\mathbf{W}_1^2$.

2.3.5$_s$ Compute the inverse 1-level Haar transform of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0),$$

and of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 0, \ldots, 0 \,|\, 1, 0, \ldots, 0).$$

Section 2.4

Example 2.4.A For $\mathbf{f} = (3, 2, -1, 4)$ and $\mathbf{g} = (2, -2, 2, -2)$, find the sum $\mathbf{f} + \mathbf{g}$, the difference $\mathbf{f} - \mathbf{g}$, and the combination $2\mathbf{f} - 3\mathbf{g}$.

Solution. We calculate that

$$\begin{aligned}
\mathbf{f} + \mathbf{g} &= (3+2, 2-2, -1+2, 4-2) = (5, 0, 1, 2) \\
\mathbf{f} - \mathbf{g} &= (3-2, 2+2, -1-2, 4+2) = (1, 4, -3, 6) \\
2\mathbf{f} - 3\mathbf{g} &= (6-6, 4+6, -2-6, 8+6) = (0, 10, -8, 14).
\end{aligned}$$

Example 2.4.B For $\mathbf{f} = (2, 2, 4, 6, -2, -2, -2, 0)$ find the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

Solution. The trend $\mathbf{a}^1$ and fluctuation $\mathbf{d}^1$ satisfy

$$\begin{aligned}
\mathbf{a}^1 &= (2\sqrt{2}, 5\sqrt{2}, -2\sqrt{2}, -\sqrt{2}) \\
\mathbf{d}^1 &= (0, -\sqrt{2}, 0, -\sqrt{2}).
\end{aligned}$$

Hence

$$\begin{aligned}
\mathbf{A}^1 &= (2, 2, 5, 5, -2, -2, -1, -1) \\
\mathbf{D}^1 &= (0, 0, -1, 1, 0, 0, -1, 1)
\end{aligned}$$

and, as a check, we observe that $\mathbf{A}^1 + \mathbf{D}^1 = \mathbf{f}$.

Example 2.4.C For $\mathbf{f} = (2, 2, 4, 6, -2, -2, -2, 0)$ find the second averaged signal $\mathbf{A}^2$ and the second detail signal $\mathbf{D}^2$.

Solution. We found in the previous example that $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, -2\sqrt{2}, -\sqrt{2})$. The second trend is then $\mathbf{a}^2 = (7, -3)$, and the second fluctuation is $\mathbf{d}^2 = (-3, -1)$. Therefore,

$$\begin{aligned}
\mathbf{A}^2 &= \left(\frac{7}{2}, \frac{7}{2}, \frac{7}{2}, \frac{7}{2}, \frac{-3}{2}, \frac{-3}{2}, \frac{-3}{2}, \frac{-3}{2}\right) \\
\mathbf{D}^2 &= \left(\frac{-3}{2}, \frac{-3}{2}, \frac{3}{2}, \frac{3}{2}, \frac{-1}{2}, \frac{-1}{2}, \frac{1}{2}, \frac{1}{2}\right)
\end{aligned}$$

© 2008 by Taylor & Francis Group, LLC

and, as a check, we observe that $\mathbf{A}^2 + \mathbf{D}^2 = \mathbf{A}^1$.

Example 2.4.D [Figure 2.3] The graphs in Figure 2.3 were created by graphing the function in Example 2.1.C, and then choosing *Series/Wavelet* and selecting *Haar* for the wavelet, using 10 for the number of Levels, and choosing *Ascending terms* for the Series type. You then enter successively 1, 2, 4,..., 512 for the number of terms to use (in the text box to the right of *Ascending terms*). For 1, we get the plot of $\mathbf{A}^{10}$, for 2 we get the plot of $\mathbf{A}^9$, for 4 we get the plot of $\mathbf{A}^8$, ..., for 512 we get the plot of $\mathbf{A}^1$.

2.4.1 For the following signals, $\mathbf{f}$ and $\mathbf{g}$, compute their sum $\mathbf{f}+\mathbf{g}$, their difference $\mathbf{f}-\mathbf{g}$, and the combination $3\mathbf{f}-2\mathbf{g}$.

(a)$_\mathbf{s}$ $\mathbf{f} = (2, 3, 2, 4)$, $\mathbf{g} = (1, 2, -1, 3)$

(b) $\mathbf{f} = (4, 2, 1, 1, 0, 0)$, $\mathbf{g} = (-1, 2, 1, 2, -1, 3)$

(c)$_\mathbf{s}$ $\mathbf{f} = (-1, -1, 1, 1, 1, 1)$, $\mathbf{g} = (1, 2, 0, 1, -1, 1)$

(d) $\mathbf{f} = (1, 1, 2, 1)$, $\mathbf{g} = (1, 2, -3, -4)$

2.4.2$_\mathbf{s}$ For each of the signals in problem 2.1.1, compute the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

2.4.3 Find expressions for the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$ in terms of the values of $\mathbf{f} = (f_1, f_2, \ldots, f_N)$.

2.4.4 For each of the following signals, compute the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

(a)$_\mathbf{s}$ $(2, 1, 3, -1, 2, 4, 3, 4)$

(b) $(1, 2, 3, 4, 5, 6, 7, 8)$

(c)$_\mathbf{s}$ $(1, 2, -1, -1, 4, 3, 2, 2)$

(d) $(9, 4, 1, 0, 0, 1, 4, 9)$

2.4.5 Express the 2^{nd} averaged signal $\mathbf{A}^2$ and the 2^{nd} detail signal $\mathbf{D}^2$ in terms of the values of $\mathbf{f} = (f_1, f_2, \ldots, f_N)$.

2.4.6$_\mathbf{s}$ For each of the signals in problem 2.4.4, compute the 2^{nd} averaged signal $\mathbf{A}^2$ and the 2^{nd} detail signal $\mathbf{D}^2$.

Section 2.5

Example 2.5.A Produce graphs like the ones shown in Figure 2.4 for 1024 samples of the following signal:

$$f(x) = 3(1 < x < 4) - 2(5 < x < 8) + 4(10 < x < 18)$$

over the interval $[0, 20]$. What threshold should be used to retain 99.99% of the energy? What compression ratio does this produce, and what is the maximum error between the original signal and the compressed signal?

Solution. Choosing *New 1-dim* and then choosing *Graph/Plot*, we plot the formula above over the interval of type $[0, L]$ setting $L = 20$. This produces the graph

© 2008 by Taylor & Francis Group, LLC

shown in Figure 2.8(a) (after we choose *View/Display style* and select a *Blank* Grid style and a *Dots* Plot style). Selecting *Transform/Wavelet* and choosing *Haar* with 10 levels, we obtain the Haar transform shown in Figure 2.8(b) (after choosing *View/Display style* and changing the Y-range to $-20, 20$). By *right*-clicking on the Haar transform graph and choosing *Sort magnitudes* and then right-clicking again and choosing *Energy graph* and entering 2 for the graph number (leaving the *percentage* box checked), we produce the graph shown in Figure 2.8(c) (after right-clicking and choosing *Clip* and clipping graph 3). Finally, by choosing *Analysis/Trace* from the menu of the window containing the Haar transform along with its sorted magnitudes and energy map, we find by tracing on the three graphs that 100% of the energy is used when the Index is 45 (i.e., 46 values used because in FAWAV the Index counter for arrays of data is initialized at 0 rather than 1); this represents $1024/46 \approx 22$ to 1 compression.

We also find, using the *Trace* procedure, that 99.99% of the energy is used with a threshold of 0.5. By graphing the function `g1(x)(abs(g1(x))>0.5)` we produce a fourth graph that is a thresholded transform. Then by choosing *Transform/Wavelet*, selecting Haar *with the Inverse box checked,* and entering 4 for the graph number, we plot an approximation of the given step function. We then right-click on the graph of this approximation, choose *Copy,* return to the window with the original function displayed and right-click on its graph followed by selecting *Paste*. This pastes the approximate function data into the window, for comparison with the original graph. Selecting *Analysis/Norm difference* and using the default values (*Sup-norm,* graphs 1 and 2, and *absolute*) we find that the maximum error (in magnitude) between the original signal and the approximation is 0.1094. (*Note:* since the original signal consists of integer values, by graphing the function `gri(g2(x)+1/2)` we round the signal's values to their nearest integers, thus producing a signal that is equal to the original signal at all values.)

Example 2.5.B [Figure 2.4] To produce Figure 2.4(a) you graph

```
(2<=x<4)-(6<=x<9)+2[14<=x<18]-2(10<= x< 11)
```

over the interval $[0, 20]$ using 1024 points. Figure 2.4(b) is obtained using a 10-level Haar transform. To get Figure 2.4(c) you right-click on the graph of the Haar transform and select *Sort magnitudes,* then right-click again and select *Energy graph.* You then clip out this third graph, to produce the energy graph shown in Figure 2.4(c). Figure 2.4(d) was obtained by selecting *Analysis/Trace* and using the tracing tool to determine that the 52 highest magnitude transform values account for 100% of the energy of the signal. Then, by returning to the window containing the original signal, selecting *Series/Wavelet* and performing a Haar series[4] with Series type *Highest mag. coefficients*, and entering 52 for the number of coefficients, we obtain the graph shown in Figure 2.4(d).

Example 2.5.C [Figure 2.5] Figure 2.5 is produced in the same way as Figure 2.4 (see preceding example), except that 4096 points are used and the following function

```
40x^2(1-x)^4cos(12pi x)[0<x<1)+{40(x-1)^2(2-x)^8cos(48pi x)
   +80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

[4]See the **Remark** on the next page.

© 2008 by Taylor & Francis Group, LLC

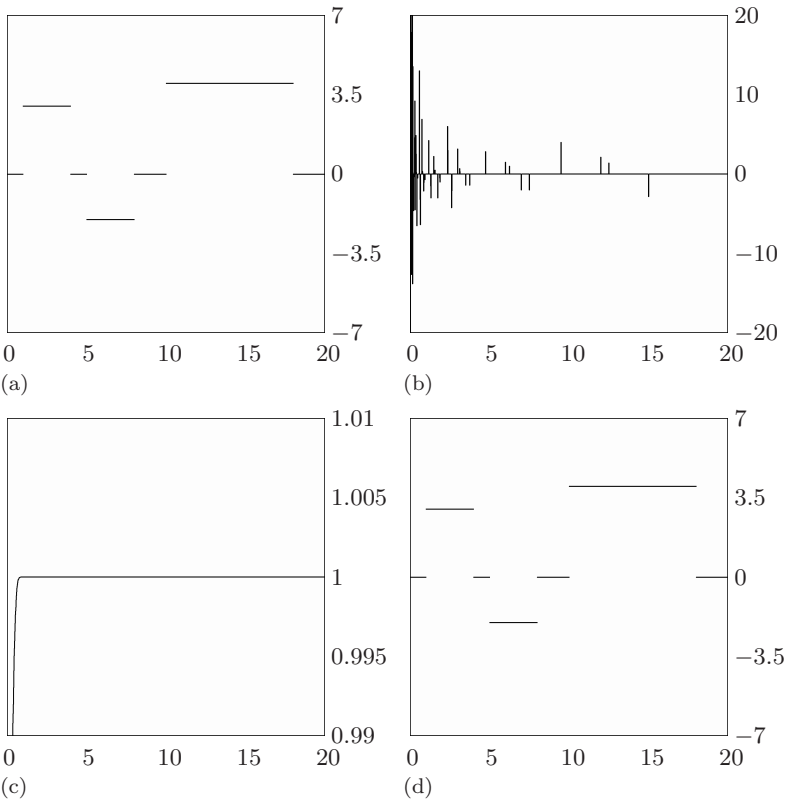


FIGURE 2.8
(a) Signal, (b) 10-level Haar transform, (c) energy map of Haar transform, (d) 22:1 compression of Signal, 100% of energy.

is graphed over $[0, 2]$ to produce the initial signal.

2.5.1$^{c}_{s}$ Produce graphs like the ones shown in Figure 2.4 for 1024 samples of the following signal:

$$f(x) = 2[1 < x < 2] - 3[3 < x < 5] + [6 < x < 7] + 4[8 < x < 9]$$

over the interval $[0, 10]$. What threshold should be used to retain 99.99% of the energy? What compression ratio does this produce, and what is the maximum error between the original signal and the compressed signal? [*Note:* maximum error is computed using the *sup-norm* method described below in Example 2.5.D.]

2.5.2^{c} Repeat problem 2.5.1 for each of the following:

(a)$_{s}$ $f(x) = x(10 - x)$

(b) $f(x) = 2[2 < x < 4] - 2[5 < x < 7] + 2[8 < x < 9]$

(c) $f(x) = 2[2 < x < 4] - x[5 < x < 7] + 2[8 < x < 9]$

Remark. The next three exercises deal with Haar wavelet series. A *wavelet series* is a convenient term for the following three step process:

© 2008 by Taylor & Francis Group, LLC

1. Compute a transform, either wavelet or wavelet packet.
2. Modify the transform values from Step 1.
3. Compute the inverse transform of the modified values from Step 2.

There are a number of ways to carry out the modification of the wavelet transform values in Step 2. The most common method is to threshold the values. When a threshold method is used, we say that the three step process produces a *thresholded series.* To produce a thresholded series with FAWAV, you begin by selecting *Series* from the menu. You then select either *Wavelet* or *Wavelet packet* to specify which of the two types of transform will be used. A dialog box will then appear on the right side of the window, and you select *Threshold* as the method for the series. Another type of series is *Highest mag.,* which modifies the transform values by using only a specified number of the highest magnitude transform values (setting all others equal to zero).

Example 2.5.D For the function in Example 2.5.A, compute the Haar series using the 30 highest magnitude values. What is the sup-norm difference between the function and its wavelet series? What is the *Sup-norm difference* over the interval $[2, 3]$?

Solution. We plot the function as described in the solution of Example 2.5.A. Selecting *Series/Wavelet* and then *Haar* for the transform and *Highest mag. coefficients* with 30 as the number, we obtain an approximation of the original signal. The *sup-norm difference* between the two signals is 1.98. To find the sup-norm difference over the interval $[2, 3]$ we choose *View/Display style* to change the X-interval to $[2, 3]$. The graphs are then displayed over the interval $[2, 3]$ and computing a sup-norm difference between them yields 0.0117.

2.5.3$^{\mathbf{c}}_{\mathbf{s}}$ For the functions in problem 2.5.2 (a) and (b), compute the Haar series using the 50 highest magnitude values. Which function is best approximated by such a Haar series? Why?

2.5.4$^{\mathbf{c}}$ For each of the signals in problem 2.5.2(b) and (c), compute the Haar series graphed in problem 2.5.3 using the highest 30 magnitude values. What is the maximum error between the original function and the Haar series over the interval $[2.5, 3.5]$, and over the interval $[5.5, 6.5]$? Why are the errors very similar over $[2.5, 3.5]$, but not over $[5.5, 6.5]$?

2.5.5 Suppose a Haar transform $(\mathbf{a}^k \,|\, \mathbf{d}^k \,|\, \ldots \,|\, \mathbf{d}^1)$ is thresholded, producing a signal $(\widetilde{\mathbf{a}}^k \,|\, \widetilde{\mathbf{d}}^k \,|\, \ldots \,|\, \widetilde{\mathbf{d}}^1)$, and then an inverse Haar transform is performed on the thresholded signal, producing a Haar series. Find an expression for this Haar series in terms of scaling signals and wavelets.

Section 2.6

Example 2.6.A [Figure 2.6] To get Figure 2.6(a), you graph

```
0.1ran(x)+(2<=x<4)-(6<=x<9)+2[14<=x<18]-2(10<= x< 11)
```

over $[0, 20]$ using 1024 points. Figure 2.6(b) was obtained in the following way. First, a Haar transform with 10 levels was performed and then the Y-interval changed to

© 2008 by Taylor & Francis Group, LLC

$[-4, 4]$. Second, by using the *Trace* tool, or by right-clicking on the Haar transform and selecting *Display cursor coordinates* to use the mouse to scan over the graph box with a readout of x- and y-values, we determined that 0.2 was a good threshold for removing noise. (The horizontal graphs in Figure 2.6(b) were obtained by plotting `0.2` and `-0.2` with the *Auto-fit* option unchecked.) To get the thresholded transform shown in Figure 2.6(c), we plotted the function

```
g1(x)(abs(g1(x))>c) \c=0.2
```

Finally, to obtain the denoised signal in Figure 2.6(d) we performed an inverse Haar transform on the thresholded transform. The RMS Error was computed by selecting *Analysis/Norm difference* and choosing a *Power norm* with power 2, using the *Absolute* and *Normalized* choices.

Example 2.6.B [Figure 2.7] The graphs in Figure 2.7 were obtained in the same way as Figure 2.6, except that the initial noisy signal was graphed using the formula

```
0.1ran(x)+40x^2(1-x)^4cos(12pi x)[0<x<1)
  +{40(x-1)^2(2-x)^8cos(48pi x)+80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

over the interval $[0, 2]$ using 4096 points, and 12 levels were used for the Haar transform.

2.6.1^{c} Graph the random noise signal, $f(x) = \text{rang}(x)$, over the interval $[0, 1]$ using 8192 points. Then play the sound generated by this signal, using *Graph/Audio* in FAWAV, with a sampling rate of 8820, a bit rate of 16 bits, and volume level of 10000. What does this random noise sound like?

2.6.2$^{c}_{s}$ Perform a 1-level Haar transform of the signal in problem 2.6.1. What does this Haar transform look like, and what does it sound like (when it is played using the same parameter values as in 2.6.1)?

2.6.3^{c} Using the Threshold Method, denoise each of the following signals (in each case, use 1024 points and $[0, 10]$ as interval):

(a)$_{s}$ $f(x) = 40[2 < x < 4] - 60[5 < x < 7] + 80[8 < x < 9] + 10\,\text{rang}(0)$

(b) $f(x) = 4\sin(2\pi x) + 0.5\,\text{rang}(0)$

(c) $f(x) = 40[2 < x < 4] + 8x[5 < x < 7] + 40[8 < x < 9] + 10\,\text{rang}(0)$

(d)$_{s}$ $f(x) = [40\cos(2\pi x)](2 < x < 6) + 10\,\text{rang}(0)$

Which denoisings would you deem to be the most successful, and why?

2.6.4 Explain the cause of the very ragged, jumpy appearance of the denoised signal in Figure 2.7(d).

© 2008 by Taylor & Francis Group, LLC

Chapter 3

Daubechies wavelets

It is hardly an exaggeration to say that we will introduce almost as many analysis algorithms as there are signals... signals are so rich and complex that a single analysis method... cannot serve them all.

—Yves Meyer[1]

In this chapter we describe a large collection of wavelet transforms discovered by Ingrid Daubechies. The Daubechies wavelet transforms are defined in the same way as the Haar wavelet transform—by computing running averages and differences via scalar products with scaling signals and wavelets. For the Daubechies wavelet transforms, the scaling signals and wavelets have slightly longer supports, i.e., they produce averages and differences using just a few more values from the signal. This slight change, however, provides a tremendous improvement in the capabilities of these new transforms, providing us with a set of powerful tools for signal processing.

3.1 The Daub4 wavelets

There are many Daubechies transforms, but they are all very similar. In this section we shall concentrate on the simplest one, the Daub4 wavelet transform. The Daub4 wavelet transform is defined in essentially the same way as the Haar wavelet transform. If a signal $\mathbf{f}$ has an even number N of values, then the 1-level Daub4 transform is the mapping $\mathbf{f} \stackrel{\mathbf{D}_1}{\longmapsto} (\mathbf{a}^1 \mid \mathbf{d}^1)$ from the signal $\mathbf{f}$

[1]Meyer's quote is from [1].

© 2008 by Taylor & Francis Group, LLC

to its first trend subsignal $\mathbf{a}^1$ and first fluctuation subsignal $\mathbf{d}^1$. Each value a_m of the trend $\mathbf{a}^1 = (a_1, \ldots, a_{N/2})$ is equal to a scalar product:

$$a_m = \mathbf{f} \cdot \mathbf{V}_m^1 \tag{3.1}$$

of $\mathbf{f}$ with a 1-level *scaling signal* $\mathbf{V}_m^1$. Likewise, each value d_m of the fluctuation $\mathbf{d}^1 = (d_1, \ldots, d_{N/2})$ is equal to a scalar product:

$$d_m = \mathbf{f} \cdot \mathbf{W}_m^1 \tag{3.2}$$

of $\mathbf{f}$ with a 1-level *wavelet* $\mathbf{W}_m^1$. We shall define these Daub4 scaling signals and wavelets in a moment, but first we shall briefly describe the higher level Daub4 transforms.

The Daub4 wavelet transform, like the Haar transform, can be extended to multiple levels as many times as the signal length can be divided by 2. This extension is similar to the way the Haar transform is extended, i.e., by applying the 1-level Daub4 transform $\mathbf{D}_1$ to the first trend $\mathbf{a}^1$. This produces the mapping $\mathbf{a}^1 \overset{\mathbf{D}_1}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2)$ from the first trend subsignal $\mathbf{a}^1$ to a second trend subsignal $\mathbf{a}^2$ and second fluctuation subsignal $\mathbf{d}^2$. The 2-level Daub4 transform $\mathbf{D}_2$ is then defined by the mapping $\mathbf{f} \overset{\mathbf{D}_2}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)$. For example, we show in Figure 3.1(b) the 2-level Daub4 transform of the signal shown in Figure 2.1(a). As with the Haar transform, the values of the second trend $\mathbf{a}^2$ and second fluctuation $\mathbf{d}^2$ can be obtained via scalar products with second-level scaling signals and wavelets. Likewise, the definition of a k-level Daub4 transform is obtained by applying the 1-level transform to the preceding level trend subsignal, just like in the Haar case. And, as in the Haar case, the values of the k-level trend subsignal $\mathbf{a}^k$ and fluctuation subsignal $\mathbf{d}^k$ are obtained as scalar products of the signal with k-level scaling signals and wavelets.

The difference between the Haar transform and the Daub4 transform lies in the way that the scaling signals and wavelets are defined. We shall first discuss the scaling signals. Let the *scaling numbers* $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be defined by

$$\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \alpha_3 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad \alpha_4 = \frac{1-\sqrt{3}}{4\sqrt{2}}. \tag{3.3}$$

Later in this chapter and Chapter 5, we shall describe how these scaling numbers were obtained. Using these scaling numbers, the 1-level Daub4 scaling

© 2008 by Taylor & Francis Group, LLC

signals are

$$\begin{aligned}
\mathbf{V}_1^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
\mathbf{V}_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2-1}^1 &= (0, 0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) \\
\mathbf{V}_{N/2}^1 &= (\alpha_3, \alpha_4, 0, 0, \ldots, 0, \alpha_1, \alpha_2).
\end{aligned} \tag{3.4}$$

These scaling signals are all very similar to each other. For example, each scaling signal has a support of just four time-units. Notice also that the second scaling signal $\mathbf{V}_2^1$ is just a translation by two time-units of the first scaling signal $\mathbf{V}_1^1$. Likewise, the third scaling signal $\mathbf{V}_3^1$ is a translation by four time-units of $\mathbf{V}_1^1$, and each subsequent scaling signal is a translation by a multiple of two time-units of $\mathbf{V}_1^1$. There is one wrinkle here. For $\mathbf{V}_{N/2}^1$, we would have to translate $\mathbf{V}_1^1$ by $N-2$ time-units, but since $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ has length 4 this would send α_3 and α_4 beyond the length N of the signal $\mathbf{f}$. To avoid this problem, we *wrap-around* to the start; hence $\mathbf{V}_{N/2}^1 = (\alpha_3, \alpha_4, 0, 0, \ldots, 0, \alpha_1, \alpha_2)$. The Haar scaling signals also have this property of being translations by multiples of two time-units of the first scaling signal. But, since the first Haar scaling signal has a support of just two adjacent non-zero values, there is no wrap-around effect in the Haar case.

The second level Daub4 scaling signals are produced by repeating the operations that were used on the natural basis of signals $\mathbf{V}_1^0, \mathbf{V}_2^0, \ldots, \mathbf{V}_N^0$ to generate the first level scaling signals.[2] Using this natural basis, the first level Daub4 scaling signals satisfy

$$\mathbf{V}_m^1 = \alpha_1 \mathbf{V}_{2m-1}^0 + \alpha_2 \mathbf{V}_{2m}^0 + \alpha_3 \mathbf{V}_{2m+1}^0 + \alpha_4 \mathbf{V}_{2m+2}^0 \tag{3.5a}$$

with a wrap-around defined by $\mathbf{V}_{n+N}^0 = \mathbf{V}_n^0$. Similarly, the second level Daub4 scaling signals are defined by

$$\mathbf{V}_m^2 = \alpha_1 \mathbf{V}_{2m-1}^1 + \alpha_2 \mathbf{V}_{2m}^1 + \alpha_3 \mathbf{V}_{2m+1}^1 + \alpha_4 \mathbf{V}_{2m+2}^1 \tag{3.5b}$$

with a wrap-around defined by $\mathbf{V}_{n+N/2}^1 = \mathbf{V}_n^1$. Notice that this wrap-around, or periodicity, of the first level scaling signals is implied by the wrap-around invoked above for the natural signal basis.

By examining Formula (3.5b), we can see that each second-level Daub4 scaling signal, $\mathbf{V}_m^2$, has a support of just 10 time-units, and is a translate by $4m$ time-units of $\mathbf{V}_1^2$ (if we include wrap-around). The second-level trend values are $\{\mathbf{f} \cdot \mathbf{V}_m^2\}$, and they measure trends over 10 successive values of $\mathbf{f}$, located in the same time positions as the non-zero values of $\mathbf{V}_m^2$. Hence these

[2] This natural basis of signals was defined in (2.20) on p. 17.

© 2008 by Taylor & Francis Group, LLC

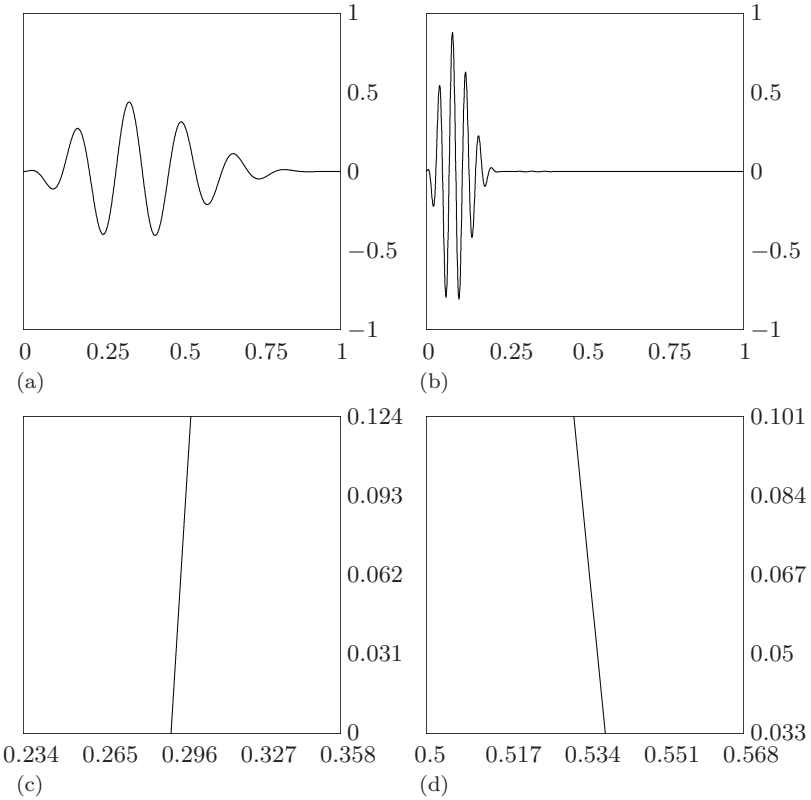


FIGURE 3.1
(a) Signal. (b) 2-level Daub4 transform. The trend $\mathbf{a}^2$ is graphed over $[0, 0.25)$, while the fluctuations $\mathbf{d}^2$ and $\mathbf{d}^1$ are graphed over $[0.25, 0.5)$ and $[0.5, 1)$, respectively. (c) and (d) Magnifications of the signal's graph in two small squares; the signal is approximately linear.

trends are measured over short time intervals that are shifts by multiples of 4 time-units of the interval consisting of the first 10 time-units. These 10-unit trends are slightly more than twice as long lasting as the trends measured by the first level scaling signals.

The k-level Daub4 scaling signals are defined by formulas similar to (3.5a) and (3.5b), but applied to the preceding level scaling signals. In Figure 3.2(a), we show some 5-level and 6-level Daub4 scaling signals. Notice that the supports of the 6-level scaling signals are about twice as long as the supports of the 5-level scaling signals. Figure 3.2(a) also illustrates the fact that each of the 5-level scaling signals is a translate of $\mathbf{V}_1^5$, and that each of the 6-level scaling signals is a translate of $\mathbf{V}_1^6$.

An important property of these scaling signals is that they all have energy 1. This is because of the following identity satisfied by the scaling numbers:

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = 1. \tag{3.6}$$

© 2008 by Taylor & Francis Group, LLC

It is clear that (3.6) implies that each 1-level scaling signal has energy 1. To see that it also implies that each k-level scaling signal has energy 1 is more difficult; we will sketch the proof at the end of the next section.

Another identity satisfied by the scaling numbers is

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = \sqrt{2}. \tag{3.7}$$

This equation says that each 1-level trend value $\mathbf{f} \cdot \mathbf{V}_m^1$ is an average of four values of $\mathbf{f}$, multiplied by $\sqrt{2}$. It can also be shown that the sum of the ten successive non-zero values of $\mathbf{V}_m^2$ is 2, which shows that each 2-level trend value $\mathbf{f} \cdot \mathbf{V}_m^2$ is an average of ten successive values of $\mathbf{f}$, multiplied by 2. Similarly, each k-level trend value $\mathbf{f} \cdot \mathbf{V}_m^k$ is an average of values of $\mathbf{f}$ over increasingly longer time intervals as k increases.

We now turn to a discussion of the Daub4 wavelets. Let the *wavelet numbers* $\beta_1, \beta_2, \beta_3, \beta_4$ be defined by

$$\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \quad \beta_2 = \frac{\sqrt{3}-3}{4\sqrt{2}}, \quad \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}}. \tag{3.8}$$

Notice that the wavelet numbers are related to the scaling numbers by the equations: $\beta_1 = \alpha_4$, $\beta_2 = -\alpha_3$, $\beta_3 = \alpha_2$, and $\beta_4 = -\alpha_1$. Using these wavelet numbers, the 1-level Daub4 wavelets $\mathbf{W}_1^1, \ldots, \mathbf{W}_{N/2}^1$ are defined by

$$\begin{aligned}
\mathbf{W}_1^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
\mathbf{W}_2^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
\mathbf{W}_3^1 &= (0, 0, 0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{W}_{N/2-1}^1 &= (0, 0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4) \\
\mathbf{W}_{N/2}^1 &= (\beta_3, \beta_4, 0, 0, \ldots, 0, \beta_1, \beta_2).
\end{aligned} \tag{3.9}$$

These wavelets are all translates of $\mathbf{W}_1^1$, with a wrap-around for the last wavelet. Each wavelet has a support of just four time-units, corresponding to the four non-zero wavelet numbers used to define them. The 1-level Daub4 wavelets satisfy

$$\mathbf{W}_m^1 = \beta_1 \mathbf{V}_{2m-1}^0 + \beta_2 \mathbf{V}_{2m}^0 + \beta_3 \mathbf{V}_{2m+1}^0 + \beta_4 \mathbf{V}_{2m+2}^0.$$

Similarly, the 2-level Daub4 wavelets are defined by

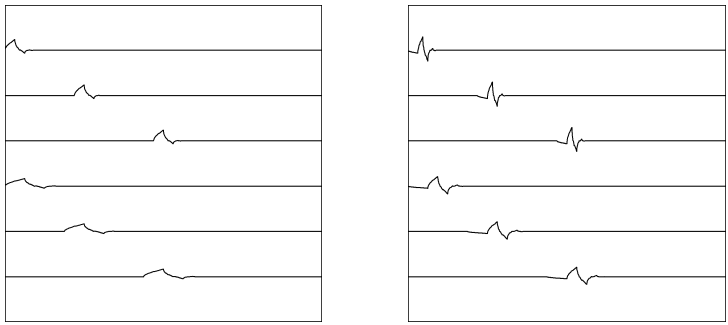
$$\mathbf{W}_m^2 = \beta_1 \mathbf{V}_{2m-1}^1 + \beta_2 \mathbf{V}_{2m}^1 + \beta_3 \mathbf{V}_{2m+1}^1 + \beta_4 \mathbf{V}_{2m+2}^1.$$

All other levels of Daub4 wavelets are defined in a similar fashion. In Figure 3.2(b) we show some of the Daub4 wavelets. It is interesting to compare them with the Daub4 scaling functions shown in Figure 3.2(a).

The Daub4 wavelets all have energy 1. This is clear for the 1-level Daub4 wavelets, since

$$\beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1. \tag{3.10}$$

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 3.2
(a) The top three signals are 5-level Daub4 scaling signals $\mathbf{V}_1^5$, $\mathbf{V}_8^5$, and $\mathbf{V}_{16}^5$. The bottom three signals are 6-level scaling signals $\mathbf{V}_1^6$, $\mathbf{V}_4^6$, and $\mathbf{V}_8^6$. (b) The top three signals are 5-level Daub4 wavelets $\mathbf{W}_1^5$, $\mathbf{W}_8^5$, and $\mathbf{W}_{16}^5$. The bottom three signals are 6-level wavelets $\mathbf{W}_1^6$, $\mathbf{W}_4^6$, and $\mathbf{W}_8^6$.

It can also be shown that all k-level Daub4 wavelets have energy 1 as well.

Each fluctuation value $d_m = \mathbf{f} \cdot \mathbf{W}_m^1$ can be viewed as a differencing operation on the values of $\mathbf{f}$ because

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 = 0. \tag{3.11}$$

Equation (3.11) is a generalization of the Haar case, where we had $1/\sqrt{2} - 1/\sqrt{2} = 0$. It also implies, as with the Haar case, that a fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^1$ will be zero if the signal $\mathbf{f}$ is constant over the support of a Daub4 wavelet $\mathbf{W}_m^1$. Much more is true, however. Not only is (3.11) true, but we also have

$$0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 = 0. \tag{3.12}$$

Equations (3.11) and (3.12), and Equation (3.7), imply the following property for the k-level Daub4 wavelet transform.

Property I. *If a signal* $\mathbf{f}$ *is (approximately) linear over the support of a* k*-level Daub4 wavelet* $\mathbf{W}_m^k$*, then the* k*-level fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^k$ *is (approximately) zero.*

For the 1-level case, Property I follows easily from Equations (3.11) and (3.12). It is more difficult to prove Property I for the k-level case, when $k > 1$, and it is for such cases that Equation (3.7) is needed.

To see why Property I is so important, we examine how it relates to sampled signals. In Figure 3.1(a) we show a signal obtained from uniformly spaced samples over the interval $[0, 1)$ of a function which has a continuous second derivative. As shown in Figure 3.1(b), the 1-level and 2-level fluctuations $\mathbf{d}^1$ and $\mathbf{d}^2$ have values that are all very near zero. This is because a large proportion of the signal consists of values that are approximately linear over a support of one of the Daub4 wavelets. For example, in Figures 3.1(c) and (d)

© 2008 by Taylor & Francis Group, LLC

we show magnifications of small squares centered at the points $(0.296, 0.062)$ and $(0.534, 0.067)$. It is clear from these figures that the signal values are approximately linear within these small squares. This is true of a large number of points on the graph of the signal and implies that many of the fluctuation values for this signal will be near zero. The basic principles of Calculus tell us that this example is typical for a signal that is sampled from a function that has a continuous second derivative. We shall discuss this point in more detail at the end of this section.

Each level Daub4 transform has an inverse. The inverse for the 1-level Daub4 transform, which maps the transform $(\mathbf{a}^1 \,|\, \mathbf{d}^1)$ back to the signal $\mathbf{f}$, is calculated explicitly as

$$\mathbf{f} = \mathbf{A}^1 + \mathbf{D}^1 \tag{3.13a}$$

with *first averaged signal* $\mathbf{A}^1$ defined by

$$\begin{aligned}\mathbf{A}^1 &= a_1\mathbf{V}_1^1 + a_2\mathbf{V}_2^1 + \cdots + a_{N/2}\mathbf{V}_{N/2}^1 \\ &= (\mathbf{f}\cdot\mathbf{V}_1^1)\mathbf{V}_1^1 + (\mathbf{f}\cdot\mathbf{V}_2^1)\mathbf{V}_2^1 + \cdots + (\mathbf{f}\cdot\mathbf{V}_{N/2}^1)\mathbf{V}_{N/2}^1 \end{aligned} \tag{3.13b}$$

and *first detail signal* $\mathbf{D}^1$ defined by

$$\begin{aligned}\mathbf{D}^1 &= d_1\mathbf{W}_1^1 + d_2\mathbf{W}_2^1 + \cdots + d_{N/2}\mathbf{W}_{N/2}^1 \\ &= (\mathbf{f}\cdot\mathbf{W}_1^1)\mathbf{W}_1^1 + (\mathbf{f}\cdot\mathbf{W}_2^1)\mathbf{W}_2^1 + \cdots + (\mathbf{f}\cdot\mathbf{W}_{N/2}^1)\mathbf{W}_{N/2}^1. \end{aligned} \tag{3.13c}$$

Formulas (3.13a) through (3.13c) are generalizations of similar formulas that we found for the Haar transform [see Section 2.4]. They are the first stage in a Daub4 MRA of the signal $\mathbf{f}$. We will not take the time at this point to prove that these formulas are correct; their proof involves techniques from the field of linear algebra. For those readers who are familiar with linear algebra, we provide a proof at the end of the next section. It is more important now to reflect on what these formulas mean.

Formula (3.13a) shows that the signal $\mathbf{f}$ can be expressed as a sum of an averaged signal $\mathbf{A}^1$ plus a detail signal $\mathbf{D}^1$. Because of Formula (3.13b) we can see that the averaged signal $\mathbf{A}^1$ is a combination of elementary scaling signals. Each scaling signal $\mathbf{V}_m^1$ is a short-lived signal, whose support consists of just four consecutive time-indices; the relative contribution of each scaling signal to $\mathbf{A}^1$ is measured by the trend value $a_m = \mathbf{f}\cdot\mathbf{V}_m^1$. Thus $\mathbf{A}^1$ is a sum of short-lived components which are multiples of the scaling signals $\mathbf{V}_m^1$. These scaling signals move across the time-axis in steps of just two time-units and live for only four time-units; *they measure short-lived trends in the signal via the trend values* $a_m = \mathbf{f}\cdot\mathbf{V}_m^1$. Likewise, the detail signal $\mathbf{D}^1$ is a combination of elementary wavelets $\mathbf{W}_m^1$. These wavelets $\mathbf{W}_m^1$ march across

© 2008 by Taylor & Francis Group, LLC

the time-axis in steps of two time-units and live for only four time-units. The relative contribution of each wavelet to $\mathbf{D}^1$ is measured by the fluctuation value $d_m = \mathbf{f} \cdot \mathbf{W}_m^1$. Since $\mathbf{D}^1 = \mathbf{f} - \mathbf{A}^1$, the sum of all of these short-lived fluctuations $\{d_m \mathbf{W}_m^1\}$ equals the difference between the signal $\mathbf{f}$ and its lower resolution, averaged, version $\mathbf{A}^1$. Because the 1-level wavelets $\{\mathbf{W}_m^1\}$ live for only four time-units and march across the time-axis in steps of two units, *they are able to detect very short-lived, transient, fluctuations in the signal.*

The inverse of the 2-level Daub4 transform is described by the formula

$$\mathbf{f} = \mathbf{A}^2 + \mathbf{D}^2 + \mathbf{D}^1 \tag{3.14a}$$

where

$$\mathbf{A}^2 = (\mathbf{f} \cdot \mathbf{V}_1^2)\mathbf{V}_1^2 + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N/4}^2)\mathbf{V}_{N/4}^2$$

$$\mathbf{D}^2 = (\mathbf{f} \cdot \mathbf{W}_1^2)\mathbf{W}_1^2 + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N/4}^2)\mathbf{W}_{N/4}^2 \tag{3.14b}$$

are the second averaged signal and second detail signal, respectively. The signal $\mathbf{D}^1$ is the first detail signal which we defined above. The second averaged signal $\mathbf{A}^2$ is a sum of components which are multiples of the scaling signals $\mathbf{V}_m^2$; these components move across the time-axis in steps of four time-units and live for ten time-units. The relative contribution of each scaling signal $\mathbf{V}_m^2$ to $\mathbf{A}^2$ is measured by the trend value $\mathbf{f} \cdot \mathbf{V}_m^2$. Since $\mathbf{A}^1 = \mathbf{A}^2 + \mathbf{D}^2$, the second detail signal $\mathbf{D}^2$ provides the details needed to produce the first averaged signal from the second averaged signal. This second detail signal $\mathbf{D}^2$ is a combination of the wavelets $\mathbf{W}_m^2$, which move across the time-axis in steps of four time-units and live for ten time-units. The relative contribution of each wavelet $\mathbf{W}_m^2$ to $\mathbf{D}^2$ is measured by the fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^2$. Like the 1-level wavelets, the 2-level wavelets are also able to detect transient fluctuations in a signal, but their supports are 10 units long instead of 4 units long. Hence the scale on which the 2-level wavelets measure fluctuations is slightly more than twice as long as the scale on which the 1-level wavelets measure fluctuations.

Further levels of the Daub4 transform are handled in a like manner. The k-level Daub4 transform has an inverse that produces the following MRA of the signal $\mathbf{f}$:

$$\mathbf{f} = \mathbf{A}^k + \mathbf{D}^k + \cdots + \mathbf{D}^2 + \mathbf{D}^1.$$

The formulas for $\mathbf{A}^k$ and $\mathbf{D}^k$ are (for $N_k = N/2^k$):

$$\mathbf{A}^k = (\mathbf{f} \cdot \mathbf{V}_1^k)\mathbf{V}_1^k + \cdots + (\mathbf{f} \cdot \mathbf{V}_{N_k}^k)\mathbf{V}_{N_k}^k$$

and

$$\mathbf{D}^k = (\mathbf{f} \cdot \mathbf{W}_1^k)\mathbf{W}_1^k + \cdots + (\mathbf{f} \cdot \mathbf{W}_{N_k}^k)\mathbf{W}_{N_k}^k.$$

In Figure 3.3 we show a Daub4 MRA of the same signal that we analyzed in Chapter 2 using a Haar MRA (see Figure 2.3 on p. 21). It is interesting to

© 2008 by Taylor & Francis Group, LLC

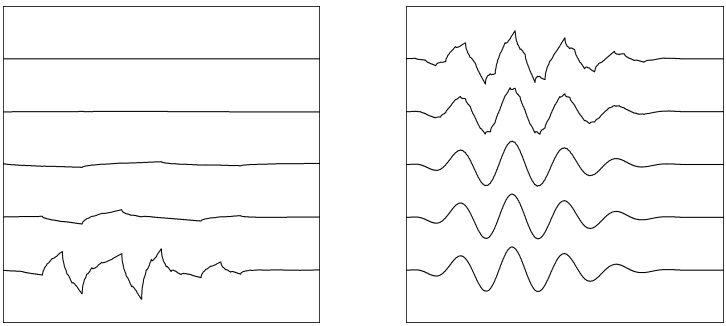
compare these two MRAs. The Daub4 MRA appears to be the superior one; it converges more quickly towards the original signal. The Daub4 averaged signals $\mathbf{A}^3$ through $\mathbf{A}^1$ all appear to be equally close approximations of the original signal. This is due to the fact that the values of the first and second Daub4 fluctuation subsignals $\mathbf{d}^1$ and $\mathbf{d}^2$ are so small [see Figure 3.1(b)] that they can be neglected without losing much detail. Likewise, the third Daub4 fluctuation subsignal $\mathbf{d}^3$ has negligible values. The corresponding Daub4 detail signals $\mathbf{D}^1$, $\mathbf{D}^2$, and $\mathbf{D}^3$ contribute very little detail to the signal; hence $\mathbf{f} \approx \mathbf{A}^3$ is a very good approximation. Another advantage of the Daub4 MRA is that the jumpy, or clumpy, appearance of the Haar averaged signals does not appear in the Daub4 averaged signals.

3.1.1 Remarks on small fluctuation values*

In discussing Property I above, we showed by means of an example that it applies to sampled signals when the analog signal has a continuous second derivative over the support of a Daub4 wavelet. That is, we assume that the signal $\mathbf{f}$ has values satisfying $f_n = g(t_n)$ for $n = 1, 2, \ldots, N$, and that the function g has a continuous second derivative over the support of a Daub4 wavelet. For simplicity we shall assume that this is a 1-level wavelet, say $\mathbf{W}^1_m$. We can then write

$$g(t_{2m-1+k}) = g(t_{2m-1}) + g'(t_{2m-1})(kh) + \mathcal{O}(h^2) \tag{3.15}$$

where $\mathcal{O}(h^2)$ stands for a quantity that is a bounded multiple of h^2. The number h is the constant step-size $h = t_{n+1} - t_n$, which holds for each n.



(a) (b)

FIGURE 3.3
Daub4 MRA of the signal shown in Figure 2.1(a) on p. 9. The graphs are of the 10 averaged signals $\mathbf{A}^{10}$ through $\mathbf{A}^1$. Beginning with $\mathbf{A}^{10}$ on the top left down to $\mathbf{A}^6$ on the bottom left, then $\mathbf{A}^5$ on the top right down to $\mathbf{A}^1$ on the bottom right. Compare with Figure 2.3 on p. 21.

© 2008 by Taylor & Francis Group, LLC

Making use of Equation (3.15), and Equations (3.11) and (3.12), we find that

$$\begin{aligned}\mathbf{f}\cdot\mathbf{W}_m^1 &= g(t_{2m-1})\{\beta_1+\beta_2+\beta_3+\beta_4\}\\ &\quad + g'(t_{2m-1})h\{0\beta_1+1\beta_2+2\beta_3+3\beta_4\}+\mathcal{O}(h^2)\\ &= \mathcal{O}(h^2).\end{aligned}$$

Thus $\mathbf{f}\cdot\mathbf{W}_m^1 = \mathcal{O}(h^2)$. This illustrates Property I, since h is generally much smaller than 1 and consequently h^2 is very tiny indeed. Our discussion also shows why the Daub4 transform generally produces much smaller fluctuation values than the Haar transform does, since for the Haar transform we typically only have $\mathbf{f}\cdot\mathbf{W}_m^1 = \mathcal{O}(h)$, which is generally much larger than $\mathcal{O}(h^2)$.

3.2 Conservation and compaction of energy

Like the Haar transform, a Daubechies wavelet transform conserves the energy of signals and redistributes this energy into a more compact form. In this section we shall discuss these properties as they relate to the Daub4 transform, but the general principles apply to all of the various Daubechies transforms.

Let's begin with a couple of examples. First, consider the signal $\mathbf{f}$ graphed in Figure 3.1(a). Using FAWAV we calculate that its energy is 509.2395. On the other hand, the energy of its 1-level Daub4 transform is also 509.2395. This illustrates the conservation of energy property of the Daub4 transform. Since the 2-level Daub4 transform consists of applying the 1-level Daub4 transform to the first trend subsignal, it follows that the 2-level Daub4 transform also conserves the energy of the signal $\mathbf{f}$. Likewise, a k-level Daub4 transform conserves energy as well.

As with the Haar transform, the Daub4 transform also redistributes the energy of the signal into a more compact form. For example, consider the signals shown in Figure 3.4. In Figure 3.4(b) we show the 2-level Daub4 transform of the signal graphed in Figure 2.2(a) on p. 13. Its cumulative energy profile is graphed in Figure 3.4(d). This cumulative energy profile shows that the 2-level Daub4 transform has redistributed almost all of the energy of the signal into the second trend subsignal, which is graphed over the first quarter of the time-interval. For comparison, we also show in Figures 3.4(a) and (c) the 2-level Haar transform and its cumulative energy profile. It is obvious from these graphs that the Daub4 transform achieves a more compact redistribution of the energy of the signal.

3.2.1 Justification of conservation of energy *

We now show why the Daub4 transform preserves the energy of each signal, and provide justifications for a couple of other statements made in the preceding section. Readers who are not familiar with linear algebra, especially matrix algebra, should skip this discussion. It will not play a major role in the material that follows, which will stress the applications of the Daubechies transforms.

© 2008 by Taylor & Francis Group, LLC

To begin, define the matrix $\mathcal{D}_N$ by

$$\mathcal{D}_N = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \alpha_3 & \alpha_4 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \alpha_1 & \alpha_2 \\ \beta_3 & \beta_4 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \beta_1 & \beta_2 \end{pmatrix}. \tag{3.16}$$

Notice that the rows of $\mathcal{D}_N$ are the first-level Daub4 scaling signals and wavelets. These scaling signals and wavelets satisfy

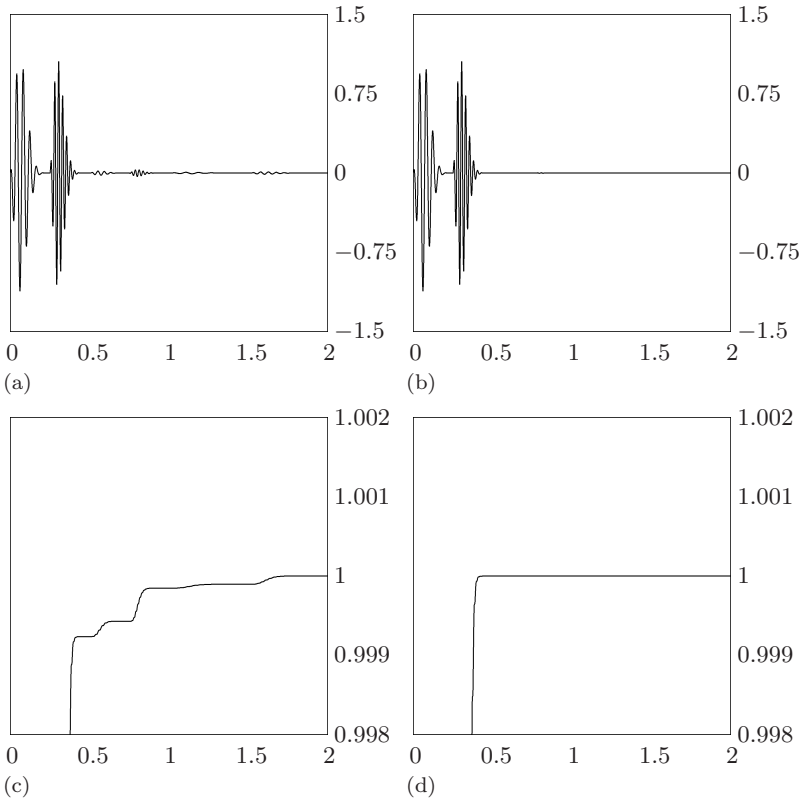


FIGURE 3.4
(a) 2-level Haar transform of signal in Figure 2.2(a) on p. 13. (b) 2-level Daub4 transform of same signal. (c) and (d) Cumulative energy profiles of the transforms in (a) and (b), respectively.

© 2008 by Taylor & Francis Group, LLC

$$\mathbf{V}_n^1 \cdot \mathbf{V}_m^1 = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m, \end{cases} \tag{3.17a}$$

$$\mathbf{W}_n^1 \cdot \mathbf{W}_m^1 = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m, \end{cases} \tag{3.17b}$$

$$\mathbf{V}_n^1 \cdot \mathbf{W}_m^1 = 0 \quad \text{all } m, n. \tag{3.17c}$$

These equations show that the rows of $\mathcal{D}_N$ form an orthonormal set of vectors, i.e., that $\mathcal{D}_N$ is an orthogonal matrix. Since $\mathcal{D}_N$ is orthogonal, it follows that

$$\mathcal{D}_N^{\mathrm{T}} \mathcal{D}_N = I_N \tag{3.18}$$

where $\mathcal{D}_N^{\mathrm{T}}$ is the transpose of $\mathcal{D}_N$ and I_N is the N by N identity matrix.

We can now show that the Daub4 transform preserves the energy of a signal $\mathbf{f}$. *These arguments will only make use of Equations* (3.17a) *to* (3.17c)*, and* (3.18)*. Therefore they will apply to all of the Daubechies transforms described in this chapter, since all of the Daubechies scaling signals and wavelets will satisfy these same equations.*[3] The matrix $\mathcal{D}_N$ will, in each case, be defined by rows consisting of the 1-level scaling signals and wavelets.

Comparing the definition of the matrix $\mathcal{D}_N$ and the definition of the 1-level Daub4 transform, we see that

$$(a_1, d_1, a_2, d_2, \ldots, a_{N/2}, d_{N/2})^{\mathrm{T}} = \mathcal{D}_N \, \mathbf{f}^{\mathrm{T}}.$$

Therefore,

$$a_1^2 + d_1^2 + \cdots + a_{N/2}^2 + d_{N/2}^2 = \left(\mathcal{D}_N \, \mathbf{f}^{\mathrm{T}}\right)^{\mathrm{T}} \left(\mathcal{D}_N \, \mathbf{f}^{\mathrm{T}}\right).$$

Furthermore, the energy $\mathcal{E}_{(\mathbf{a}^1 \,|\, \mathbf{d}^1)}$ of the 1-level Daub4 transform of $\mathbf{f}$ satisfies

$$a_1^2 + \cdots + a_{N/2}^2 + d_1^2 + \cdots + d_{N/2}^2 = \mathcal{E}_{(\mathbf{a}^1 \,|\, \mathbf{d}^1)}.$$

Since the left-hand sides of these last two equations are clearly equal, we make use of (3.18) to obtain

$$\begin{aligned} \mathcal{E}_{(\mathbf{a}^1 \,|\, \mathbf{d}^1)} &= (\mathcal{D}_N \mathbf{f}^{\mathrm{T}})^{\mathrm{T}} \, (\mathcal{D}_N \mathbf{f}^{\mathrm{T}}) \\ &= \mathbf{f} \, \mathcal{D}_N^{\mathrm{T}} \, \mathcal{D}_N \, \mathbf{f}^{\mathrm{T}} \\ &= \mathbf{f} \, \mathbf{f}^{\mathrm{T}} \\ &= \mathcal{E}_{\mathbf{f}}. \end{aligned}$$

This proves that the 1-level Daub4 transform has the conservation of energy property. As we argued above, this also shows that every level Daub4 transform conserves energy.

[3] Except in Sections 3.7 and 3.8, where energy preservation will only approximately hold for the biorthogonal wavelets discussed there.

© 2008 by Taylor & Francis Group, LLC

Another consequence of Equations (3.17a) to (3.17c) is that the Daub4 scaling signals and wavelets all have energy 1. Since the energy $\mathcal{E}_{\mathbf{f}}$ of a signal equals $\mathbf{f} \cdot \mathbf{f}$, these equations show immediately that $\mathbf{V}_m^1$ and $\mathbf{W}_m^1$ each have energy 1. To indicate why all other scaling signals and wavelets have energy 1, we will show why the wavelet $\mathbf{W}_1^2$ has energy 1. Similar arguments can be used for the other wavelets and scaling signals. Since

$$\mathbf{W}_1^2 = \beta_1 \mathbf{V}_1^1 + \beta_2 \mathbf{V}_2^1 + \beta_3 \mathbf{V}_3^1 + \beta_4 \mathbf{V}_4^1$$

we find, using (3.17a), that

$$\mathbf{W}_1^2 \cdot \mathbf{W}_1^2 = \beta_1^2 \mathbf{V}_1^1 \cdot \mathbf{V}_1^1 + \beta_2^2 \mathbf{V}_2^1 \cdot \mathbf{V}_2^1 + \beta_3^2 \mathbf{V}_3^1 \cdot \mathbf{V}_3^1 + \beta_4^2 \mathbf{V}_4^1 \cdot \mathbf{V}_4^1. \tag{3.19}$$

Notice that terms such as $\beta_n \beta_m \mathbf{V}_n^1 \cdot \mathbf{V}_m^1$ are equal to 0 when $m \neq n$; so they do not appear on the right-hand side of (3.19). Each scalar product on the right-hand side of (3.19) is 1; hence

$$\mathbf{W}_1^2 \cdot \mathbf{W}_1^2 = \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1$$

and that proves that the energy of $\mathbf{W}_1^2$ is 1.

Similar arguments can be used to show that Equations (3.13a) through (3.13c) are valid. We shall briefly indicate why this is so. Suppose that a signal $\mathbf{f}$ is defined by

$$r_1 \mathbf{V}_1^1 + \cdots + r_{N/2} \mathbf{V}_{N/2}^1 + s_1 \mathbf{W}_1^1 + \cdots + s_{N/2} \mathbf{W}_{N/2}^1 = \mathbf{f} \tag{3.20}$$

where $r_1, \ldots, r_{N/2}, s_1, \ldots, s_{N/2}$ are constants. Using (3.17a) to (3.17c), it follows that

$$r_m = \mathbf{f} \cdot \mathbf{V}_m^1, \quad s_m = \mathbf{f} \cdot \mathbf{W}_m^1 \tag{3.21}$$

for each m. In particular, if $\mathbf{f} = (0, 0, \ldots, 0)$, then $r_1 = 0, r_2 = 0, \ldots, r_{N/2} = 0$ and $s_1 = 0, s_2 = 0, \ldots, s_{N/2} = 0$. This proves that the signals

$$\mathbf{V}_1^1, \ldots, \mathbf{V}_{N/2}^1, \mathbf{W}_1^1, \ldots, \mathbf{W}_{N/2}^1$$

are linearly independent; hence they form a basis for the vector space $\mathbf{R}^N$ of all real-valued signals of length N. Consequently Equation (3.20) must hold, with unique coefficients $r_1, \ldots, r_{N/2}, s_1, \ldots, s_{N/2}$, for every signal $\mathbf{f}$. And Formula (3.21) shows that these coefficients are equal to the scalar products of $\mathbf{f}$ with the scaling signals and wavelets, exactly as described in Equations (3.13a) through (3.13c).

3.2.2 How wavelet and scaling numbers are found *

In this subsection we briefly outline how the Daub4 scaling numbers and wavelet numbers are determined. The essential features of this outline also apply to the other Daubechies wavelets that are defined in the next section.

© 2008 by Taylor & Francis Group, LLC

The constraints that determine the Daub4 scaling and wavelet numbers are Equations (3.17a) to (3.17c), (3.6), (3.7), and (3.10) through (3.12). By combining these last three equations with the requirement that $\mathbf{W}_1^1 \cdot \mathbf{W}_2^1 = 0$ from (3.17b), we obtain the following four constraining equations on the wavelet numbers:

$$\begin{aligned} \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 &= 1 \\ \beta_1 + \beta_2 + \beta_3 + \beta_4 &= 0 \\ 0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 &= 0 \\ \beta_1\beta_3 + \beta_2\beta_4 &= 0. \end{aligned}$$

These equations are sufficient for finding values for the Daub4 wavelet numbers β_1, β_2, β_3, and β_4. The scaling numbers are then determined by the equations $\alpha_1 = -\beta_4$, $\alpha_2 = \beta_3$, $\alpha_3 = -\beta_2$, and $\alpha_4 = \beta_1$.

This very brief outline only partially answers the question of how scaling numbers and wavelet numbers are found. We shall provide a more complete discussion in Chapter 5.

3.3 Other Daubechies wavelets

In this section we shall complete our introduction to the theory of the orthogonal Daubechies wavelets and wavelet transforms. In the previous two sections we described the Daub4 wavelet transform and its scaling signals and wavelets. We shall now describe the various Daub$J$ transforms for $J = 6, 8, \ldots, 20$, and the Coif$I$ transforms for $I = 6, 12, 18, 24, 30$. These wavelet transforms are all quite similar to the Daub4 transform; our treatment here will concentrate on the value of having more wavelet transforms at our disposal. There are also many more wavelet transforms—such as *spline wavelet transforms,* various types of *biorthogonal wavelet transforms,* and even more Daub$J$ and Coif$I$ transforms than the ones we describe—but we shall not try to give an exhaustive coverage of all of these transforms. Examining a few trees should give us a good feeling for the forest of wavelet transforms.[4]

Let's begin with the Daub$J$ transforms for $J = 6, 8, \ldots, 20$. The easiest way to understand these transforms is just to treat them as simple generalizations of the Daub4 transform. The most obvious difference between them is the length of the supports of their scaling signals and wavelets. For example, for the Daub6 wavelet transform, we define the scaling numbers $\alpha_1, \ldots, \alpha_6$ to be

$$\begin{aligned} \alpha_1 &= 0.332670552950083, & \alpha_2 &= 0.806891509311092, \\ \alpha_3 &= 0.459877502118491, & \alpha_4 &= -0.135011020010255, \\ \alpha_5 &= -0.0854412738820267, & \alpha_6 &= 0.0352262918857095 \end{aligned}$$

[4] We shall, however, examine two types of biorthogonal wavelet transforms in Sections 3.7 and 3.8. Biorthogonal wavelets are widely employed in image compression.

© 2008 by Taylor & Francis Group, LLC

and the wavelet numbers $\beta_1, \ldots, \beta_6$ to be

$$\beta_1 = \alpha_6,\ \beta_2 = -\alpha_5,\ \beta_3 = \alpha_4,\ \beta_4 = -\alpha_3,\ \beta_5 = \alpha_2,\ \beta_6 = -\alpha_1.$$

We then generalize the formulas in (3.4) in the following way:

$$\begin{aligned}
\mathbf{V}_1^1 &= (\alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
\mathbf{V}_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2}^1 &= (\alpha_3\, a_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, a_2)
\end{aligned} \tag{3.22}$$

with a wrap-around occurring for $\mathbf{V}_{N/2-1}^1$ and $\mathbf{V}_{N/2}^1$. The formulas in (3.22) define the first level Daub6 scaling signals. The scaling numbers satisfy (to a high degree of accuracy):

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 + \alpha_5^2 + \alpha_6^2 = 1, \tag{3.23a}$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = \sqrt{2}. \tag{3.23b}$$

Equation (3.23a) says that each scaling signal $\mathbf{V}_m^1$ has an energy of 1, while Equation (3.23b) says that the trend values $\mathbf{f} \cdot \mathbf{V}_m^1$ are averages of six successive values of $\mathbf{f}$, multiplied by $\sqrt{2}$.

The 1-level Daub6 wavelets are defined via the wavelet numbers $\beta_1, \ldots, \beta_6$ in the same manner. In fact, since all of the definitions and formulas given in the last two sections generalize in obvious ways, we shall not repeat them.

Let's consider instead what new features are exhibited by the Daub6 transform. The principal feature is that the wavelet numbers $\beta_1, \ldots, \beta_6$ satisfy the following three identities (to a high degree of accuracy):

$$\begin{aligned}
\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 &= 0, \\
0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 + 4\beta_5 + 5\beta_6 &= 0, \\
0^2\beta_1 + 1^2\beta_2 + 2^2\beta_3 + 3^2\beta_4 + 4^2\beta_5 + 5^2\beta_6 &= 0.
\end{aligned} \tag{3.24}$$

These equations, along with Equation (3.23b), imply the following property.

Property II. *If a signal* $\mathbf{f}$ *is (approximately) linear or quadratic over the support of a k-level Daub6 wavelet* $\mathbf{W}_m^k$*, then the k-level Daub6 fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^k$ *is (approximately) zero.*

Because of this property, the Daub6 transform will often produce smaller size fluctuation values than those produced by the Daub4 transform. The types of signals for which this occurs are the ones that are obtained from samples of analog signals that are at least three times continuously differentiable (at least over large portions of the analog signal). These kinds of signals are

© 2008 by Taylor & Francis Group, LLC

better approximated, over a large proportion of their values, by quadratic approximations rather than just linear approximations. Quadratic functions have curved graphs and can thereby provide superior approximations to the parts of the signal that are near to the turning points in its graph. To illustrate these ideas, consider the signal graphed at the top of Figure 3.5(a) and its 1-level Daub4 and Daub6 fluctuation subsignals graphed in the middle and at the bottom of the figure, respectively. This figure makes it clear that the Daub4 fluctuation values are significantly larger in magnitude than the Daub6 fluctuation values. It also shows that the largest magnitude Daub4 fluctuation values occur near the turning points in the graph of the signal. Similar graphs in Figure 3.5(b) illustrate the same ideas for the 3-level Daub4 and Daub6 fluctuation values.

When the goal is compression of signals, such as musical tones which often have graphs like the signal at the top of Figure 3.5(a), then the Daub6 transform can generally perform better at compressing the signal than the Daub4 transform. This is due to the larger number of Daub6 fluctuation values which can be ignored as insignificant. When the goal, however, is identifying features of the signal that are related to turning points in its graph, then the Daub4 transform can identify the location of these turning points more clearly as shown in Figure 3.5.

The Daub$J$ transforms for $J = 8, 10, \ldots, 20$, are defined in essentially the same way. The scaling numbers $\alpha_1, \ldots, \alpha_J$ satisfy

$$\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_J^2 = 1, \tag{3.25a}$$

$$\alpha_1 + \alpha_2 + \cdots + \alpha_J = \sqrt{2}. \tag{3.25b}$$

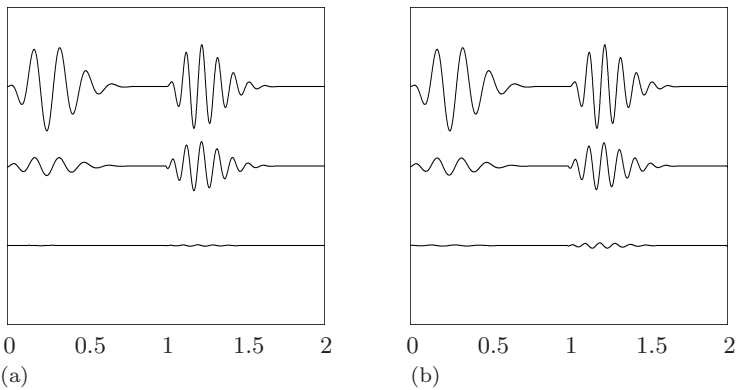


FIGURE 3.5
(a) Top: Signal. Middle: 1-level Daub4 fluctuation subsignal (multiplied by 1000 for comparison with the signal). Bottom: 1-level Daub6 fluctuation subsignal (also multiplied by 1000). (b) Similar graphs for 3-level Daub4 and Daub6 fluctuation subsignals (multiplied by 30).

© 2008 by Taylor & Francis Group, LLC

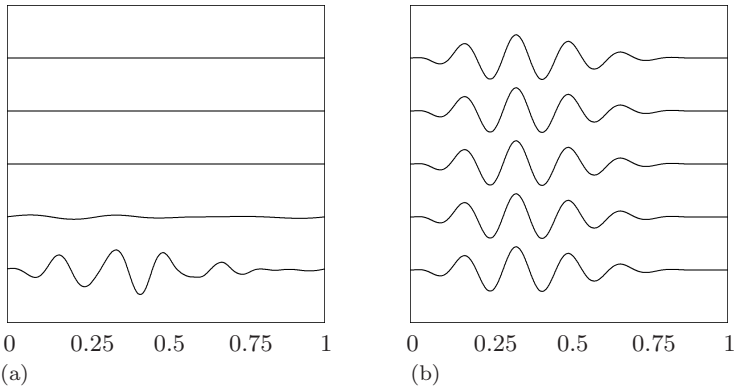


FIGURE 3.6
Daub20 MRA of the signal shown in Figure 2.1(a) on p. 9. The graphs are of the 10 averaged signals $\mathbf{A}^{10}$ through $\mathbf{A}^{1}$. Beginning with $\mathbf{A}^{10}$ on top left down to $\mathbf{A}^{6}$ on bottom left, then $\mathbf{A}^{5}$ on top right down to $\mathbf{A}^{1}$ on bottom right. Compare with Figure 2.3 on p. 21 and Figure 3.3 on p. 49.

And the wavelet numbers $\beta_1, \ldots, \beta_J$ are defined by

$$\beta_1 = \alpha_J,\ \beta_2 = -\alpha_{J-1},\ \beta_3 = \alpha_{J-2}, \ldots,\ \beta_{J-1} = \alpha_2,\ \beta_J = -\alpha_1. \qquad (3.26)$$

These wavelet numbers satisfy the following identities (we set $0^0 = 1$ to enable a single statement):

$$0^L\beta_1 + 1^L\beta_2 + \cdots + (J-1)^L\beta_J = 0, \quad \text{for } L = 0, 1, \ldots, J/2 - 1. \qquad (3.27)$$

These identities, along with (3.25b), imply the following property which is a generalization of Properties I and II above.

Property III. *If f is (approximately) equal to a polynomial of degree less than $J/2$ over the support of a k-level Daub$J$ wavelet W_m^k, then the k-level fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^k$ is (approximately) zero.*

As with Property II above, this property implies that the Daub$J$ transform will produce a large number of small fluctuation values for a signal that is sampled from a smooth, many times continuously differentiable, signal. To put it another way, we can more closely approximate (obtain a better fit to) a wider range of signals if we can use higher degree polynomials with degree less than $J/2$, and yet still expect that the Daub$J$ transform will produce large numbers of small fluctuation values. As we showed in the previous chapter, when a wavelet transform produces a large number of small fluctuation values then we can obtain very effective compression and good noise removal.

One advantage of using a Daub$J$ wavelet with a larger value for J, say $J = 20$, is that there is an improvement in the resulting MRA for smoother signals (signals sampled from analog signals having more differentiability). For example, in Figure 3.6 we show the Daub20 MRA for the same signal

© 2008 by Taylor & Francis Group, LLC

analyzed previously with Haar and Daub4 wavelets. Notice that the Daub20 MRA is superior to both of these previous multiresolution analyses, especially for the lower resolution averaged signals.

We do not mean to suggest, however, that Daub20 wavelets are always the best. For example, for Signal 1 shown in Figure 2.4(a) on p. 23, the Haar wavelets do the best job of compression and noise removal (for reasons discussed in the previous chapter). As a simple comparison of the Haar, Daub4, and Daub20 wavelets, in Table 3.1 we list the minimum number of transform values needed to capture 99.99% of the energy in Signal 1. This table shows that the Haar transform does the best job, that the Daub4 transform is worse by a factor of 2, and that the Daub20 transform is the worst of all.

Table 3.1 Comparing transforms of Signal 1

Wavelet transform	*Values for* 99.99% *energy*
Haar	51
Daub4	103
Daub20	205

The problem with Daub$J$ wavelets in terms of this signal is that these wavelets have longer supports than the Haar wavelets, all of their supports being at least twice as long, or longer, than the Haar wavelets. The Daub20 wavelets have the longest supports, with 1-level wavelets having supports of 20 time-units, and 2-level wavelets having supports of 58 time-units, and so on. Consequently, the percentage of Daub20 fluctuation values of this signal with significant energy will be high, due to the large number of Daub20 wavelets whose supports contain a point where a big jump in the signal's values occurs. A big jump in the signal's values induces corresponding jumps in the values of the scalar products that define the fluctuations, thus producing fluctuation values with significant energy.

3.3.1 Coiflets

We now turn to the description of another class of wavelets, the Coif$I$ wavelets. These wavelets are designed to maintain an extremely close match between the trend values and the original signal values. Following a suggestion of Ronald Coifman, these wavelets were first constructed by Daubechies, who called them "coiflets." All of the Coif$I$ wavelets are defined in a similar way; so we shall concentrate on the simplest case of Coif6 wavelets. The scaling numbers for the Coif6 scaling signals are the following:

$$\alpha_1 = \frac{1-\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_2 = \frac{5+\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_3 = \frac{14+2\sqrt{7}}{16\sqrt{2}},$$

$$\alpha_4 = \frac{14-2\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_5 = \frac{1-\sqrt{7}}{16\sqrt{2}}, \qquad \alpha_6 = \frac{-3+\sqrt{7}}{16\sqrt{2}}.$$

© 2008 by Taylor & Francis Group, LLC

Using these scaling numbers, the first-level Coif6 scaling signals are defined by

$$\begin{aligned}
\mathbf{V}_1^1 &= (\alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2) \\
\mathbf{V}_2^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2}^1 &= (\alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4).
\end{aligned} \tag{3.28}$$

Notice that there are wrap-arounds for $\mathbf{V}_1^1$ and $\mathbf{V}_{N/2}^1$.

The Coif6 wavelet numbers are defined by

$$\beta_1 = \alpha_6,\ \beta_2 = -\alpha_5,\ \beta_3 = \alpha_4,\ \beta_4 = -\alpha_3,\ \beta_5 = \alpha_2,\ \beta_6 = -\alpha_1 \tag{3.29}$$

and these wavelet numbers determine the first-level Coif6 wavelets as follows:

$$\begin{aligned}
\mathbf{W}_1^1 &= (\beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2) \\
\mathbf{W}_2^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0) \\
\mathbf{W}_3^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{W}_{N/2}^1 &= (\beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4).
\end{aligned} \tag{3.30}$$

As with the Coif6 scaling signals, there are wrap-arounds for the first and last Coif6 wavelets.

The Coif6 scaling numbers satisfy the following identity

$$\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 + \alpha_5^2 + \alpha_6^2 = 1 \tag{3.31}$$

which implies that each Coif6 scaling signal has energy 1. Because of (3.29), it follows that each Coif6 wavelet also has energy 1. Furthermore, the wavelet numbers satisfy

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 = 0, \tag{3.32a}$$

$$0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 + 4\beta_5 + 5\beta_6 = 0. \tag{3.32b}$$

These equations show that a Coif6 wavelet is similar to a Daub4 wavelet in that it will produce a zero fluctuation value whenever a signal is linear over its support. The difference between a Coif6 wavelet and a Daub4 wavelet lies in the properties of the scaling numbers. The Coif6 scaling numbers satisfy

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = \sqrt{2}, \tag{3.33a}$$

$$-2\alpha_1 - 1\alpha_2 + 0\alpha_3 + 1\alpha_4 + 2\alpha_5 + 3\alpha_6 = 0, \tag{3.33b}$$

$$(-2)^2\alpha_1 + (-1)^2\alpha_2 + 0^2\alpha_3 + 1^2\alpha_4 + 2^2\alpha_5 + 3^2\alpha_6 = 0. \tag{3.33c}$$

© 2008 by Taylor & Francis Group, LLC

Equation (3.33a) implies, as usual, that Coif6 trend values are averages of successive values of a signal $\mathbf{f}$ (with wrap-around when $\mathbf{f}\cdot\mathbf{V}_1^1$ and $\mathbf{f}\cdot\mathbf{V}_{N/2}^1$ are computed). The second two equations, however, are entirely new. No Daub$J$ scaling numbers satisfy any equations of these kind. These three equations have an important consequence. *When a signal consists of sample values of an analog signal, then a Coif6 transform produces a much closer match between trend subsignals and the original signal values than can be obtained with any of the DaubJ transforms.* By a close match between trends and signal values, we mean that the following approximations hold to a high degree of accuracy:

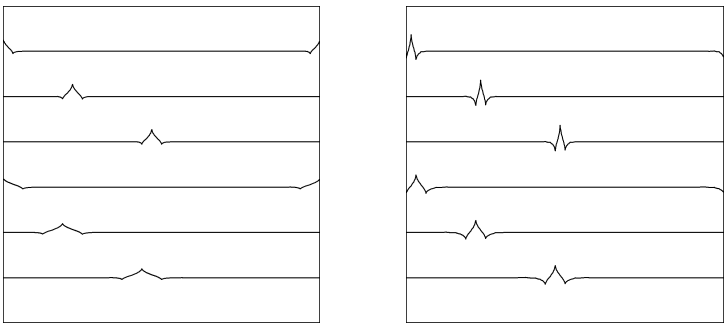
$$\mathbf{a}_m^1 \approx \sqrt{2}g(t_{2m}), \qquad \mathbf{a}_m^2 \approx 2g(t_{4m}). \tag{3.34}$$

Similar approximations will hold for higher levels, but the accuracy generally decreases as the number of levels increases.

As an example of (3.34), consider the signal graphed in Figure 3.1(a). This signal is obtained from 2^{14} sample values of the function

$$g(x) = 20x^2(1-x)^4\cos 12\pi x \tag{3.35}$$

over the interval $[0, 1)$. When a 2-level Daub4 transform is performed on this signal then we obtain the graph shown in Figure 3.1(b). A 2-level Coif6 transform looks much the same. Computing the maximum error between the 2-level Daub4 trend values and samples of $2g(4x)$ over the interval $[0, 0.25)$, we obtain 3.76×10^{-3}. The maximum error in the Coif6 case is 4.84×10^{-7}, which is much smaller. For the 1-level transforms we find that the maximum error between the first trend and samples of $\sqrt{2}g(2x)$ is 8.87×10^{-4} in the Daub4 case, and 8.59×10^{-8} in the Coif6 case. This property of trends providing close approximations of the analog signal, which is shared by all the Coif$I$ transforms, provides a useful means of interpreting the trend subsignals.



(a) (b)

FIGURE 3.7
(a) The top three signals are 5-level Coif6 scaling signals, $\mathbf{V}_1^5$, $\mathbf{V}_8^5$, and $\mathbf{V}_{16}^5$. The bottom three signals are 6-level scaling signals $\mathbf{V}_1^6$, $\mathbf{V}_4^6$, and $\mathbf{V}_8^6$. (b) The top three signals are 5-level Coif6 wavelets, $\mathbf{W}_1^5$, $\mathbf{W}_8^5$, and $\mathbf{W}_{16}^5$. The bottom three signals are 6-level wavelets $\mathbf{W}_1^6$, $\mathbf{W}_4^6$, and $\mathbf{W}_8^6$.

© 2008 by Taylor & Francis Group, LLC

Another interesting feature of Coif$I$ scaling signals and wavelets is that their graphs are nearly symmetric. For example, in Figure 3.7 we graph several Coif6 scaling signals and wavelets. Notice how these Coif6 signals (especially if we discount the wrap-around effects) are much closer to being symmetric than the Daub4 signals graphed in Figure 3.2.

3.4 Compression of audio signals

One of the fundamental applications of wavelet transforms is the compression of signals. We outlined a basic method for wavelet transform compression in Section 2.5. In that section we focused on the Haar wavelet transform; in this section we shall work with the Daubechies wavelet transforms. We shall also discuss the problem of *quantization,* which we omitted from our first treatment of compression.

Recall that the basic method of wavelet transform compression consisted of setting equal to zero all transform values whose magnitudes lie below a threshold value. The compressed version of the signal consists of the significant, non-zero, values of the transform which survived the thresholding, along with a significance map indicating their indices. Decompression consists in using the significance map and the significant transform values to reconstruct the thresholded transform, and then performing an inverse wavelet transform to produce an approximation of the original signal. Compression works well when very few, high-energy, transform values capture most of the energy of the signal. For instance, consider again Signals 1 and 2 from Section 2.5. We saw that Signal 1 can be very effectively compressed using a Haar transform. This is revealed by the energy map for its Haar transform in Figure 2.4(c) on p. 23, which shows that the Haar transform effectively captures most of the energy of Signal 1 in relatively few values.

None of the Daubechies transforms does a better job compressing Signal 1. We have already discussed why the Haar transform performs so well on this signal. In fact, the Haar transform and a related transform called the Walsh transform[5] have been used for many years as tools for compressing *piecewise constant* signals like Signal 1. We also saw, however, that Signal 2 does not compress particularly well using the Haar transform. This is explained by an examination of its energy map shown in Figure 2.5(c) on p. 25. Let's instead try compressing Signal 2 using one of the Daubechies transforms. Signal 2 consists of $4096 = 2^{12}$ points; so we will use a 12-level transform, say a Coif30 transform. In Figure 3.8, we show the results of applying a Coif30 wavelet transform compression on Signal 2. It is interesting to compare this figure with Figure 2.4 on p. 23. It is clear that the 12-level Coif30 transform compresses Signal 2 just as well as the Haar transform compresses Signal 1. In fact, by using only the top 125 highest magnitude Coif30 transform values—which can be done by choosing a threshold of 0.00425—the compressed signal

[5]The Walsh transform is described in Section 6.1.

© 2008 by Taylor & Francis Group, LLC

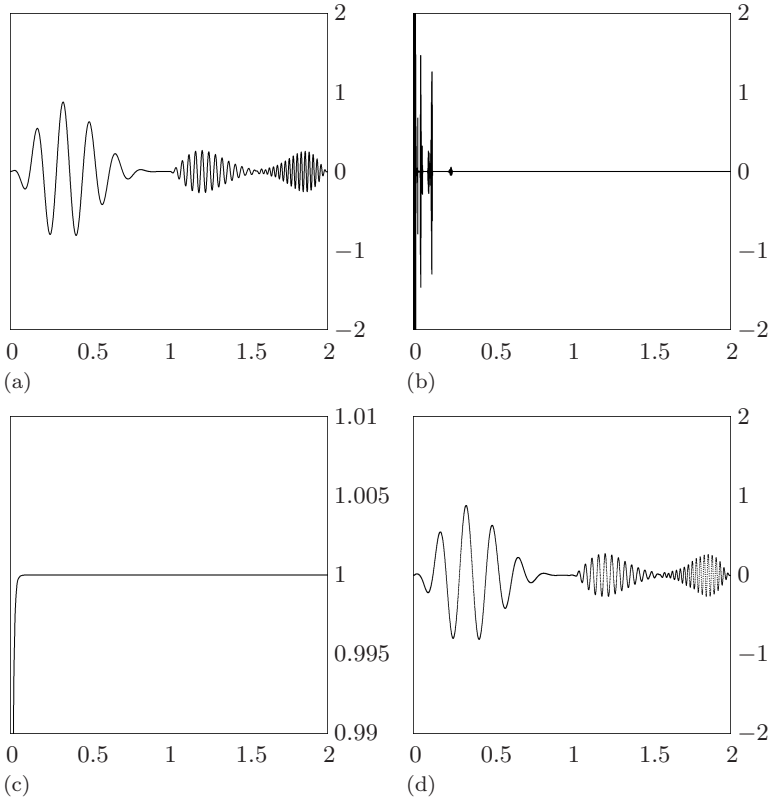


FIGURE 3.8
(a) Signal 2, 4096 values. (b) 12-level Coif30 transform. (c) Energy map of Coif30 transform. (d) 32:1 compression, 99.99% of energy of Signal 2.

captures 99.99% of the energy of Signal 2. This compressed signal is shown in Figure 3.8(d). Since $4096/125 \approx 32$, the compressed signal achieves a 32:1 compression ratio. Here we are ignoring issues such as quantization and compression of the significance map. We now turn to a brief discussion of these deeper issues of compression; this initial treatment will be expanded upon in the next section.

3.4.1 Quantization and the significance map

A digital audio signal typically consists of integer values that specify *volume levels.* The two most common ranges of volume levels are either $256 = 2^8$ volume levels, which require 8 bits to describe, or $65536 = 2^{16}$ volume levels, which require 16 bits to describe. An analog audio signal is *quantized* by a mapping from the recorded volume level to one of these two ranges of volume levels. Therefore, a discrete audio signal of length N will be initially described by either $8N$ or $16N$ bits, depending on which range of volume levels is used

© 2008 by Taylor & Francis Group, LLC

for recording. The 8-bit range is frequently used for voices in telephone transmission, where high fidelity is sacrificed for speed of transmission in order to accommodate the large number of signals that must be transmitted. The 16-bit range is frequently used for music, where high fidelity is most valued.

The most commonly employed quantization method for sampled analog signals is *uniform scalar quantization.* This method simply divides the range of volume levels into a fixed number of uniform width subintervals and rounds each volume level into the midpoint of the subinterval in which it lies. For instance, in Figure 3.9(a), we show a simple uniform scalar quantization map that encodes volume levels using 4 bits. The volume interval $[-24, 21]$ is divided into $16 = 2^4$ equal width subintervals, with all volumes that are below -24 being truncated to -24 and all volumes that are greater than 21 being truncated to 21. These 16 volume levels can be encoded as shown in Table 3.2. The asymmetry in this uniform quantization decreases as more subintervals, i.e., more bits, are used.

Table 3.2 4-BIT ENCODING

Volume level	*Encoding*
-24	1111
-21	1110
⋮	⋮
-1	1000
0	0000
1	0001
⋮	⋮
18	0110
21	0111

In order to take into account the number of bits used per point (bpp), which is either 8 bpp or 16 bpp in a quantized audio signal, we must also quantize the transform coefficients. That is, we must use only a finite number of bits to describe each transform value, and the number of bpp must be significantly less for the compressed signal than for the original signal.

As an initial example of handling quantization, consider again Signal 2 shown in Figure 3.8(a). This signal was generated from 4096 uniform samples of an analog signal. If this signal is uniformly scalar quantized with 16 bpp and played as an audio signal at a rate of 8820 samples per second,[6] multiplying its volume by a factor of 32000, then the resulting sound resembles two low notes played on a clarinet. The 16-bit quantized version of the signal has a graph that is almost identical to the one shown in Figure 3.8(a). If a Coif30 transform is performed on this quantized signal, then a transform is produced which is virtually indistinguishable from the graph shown in Figure 3.8(b).

[6]Volume levels are sent to the sound system at a rate of 8820 values per second.

© 2008 by Taylor & Francis Group, LLC

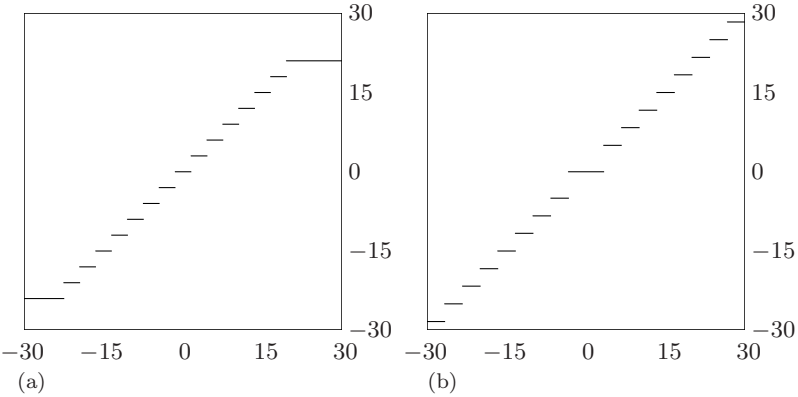


FIGURE 3.9
(a) Uniform, scalar quantization. (b) Uniform quantization, with a dead-zone containing the 0-level. The quantized values outside the dead-zone can be encoded using 4 bits.

To quantize this transform, we proceed as follows. The quantization map used is similar to the one shown in Figure 3.9(b). This is called *uniform quantization with a dead-zone.* The values in the subinterval $(-T, T)$ are the insignificant values whose magnitudes lie below a threshold value of T. Since these values will not be transmitted they are not encoded by the quantization. The remainder of the range of transform values lies in the two intervals $[-M, -T]$ and $[T, M]$, where M is the maximum for all the magnitudes of the transform values. These two intervals are divided into uniform width subintervals and each transform value is rounded into the midpoint of the subinterval containing it. For Signal 2, the value of M is 9.4, and a threshold value of $T = 9.4/2^7$ results in only 100 significant values. These significant values can then be encoded using 8 bits, 7 bits for the levels of magnitude and 1 bit for signs. As can be seen from Figure 3.8(b), the significant values of the transform lie in the interval $[0, 0.25)$. In fact, the significant values of the quantized transform lie among the first 256 values. Consequently the bits of value 1 in the significance map lie only within the first 256 bits. Therefore, this significance map can be transmitted in a very compressed form by just transmitting the first 256 bits, letting the decompression program supply the remaining bits, which are all 0. The result is that

$$\frac{256 + 8 \cdot 100}{4096} \approx 0.27 \text{ bpp}$$

are needed to transmit the compressed signal. This represents a compression ratio of 60:1. Even more importantly, when the decompressed signal is played, the resulting sound is indistinguishable from the original signal's sound. This is known as *perceptually lossless* compression.

This example was meant to illustrate the basic idea of quantization and its relation to wavelet transform compression of audio signals. In the next section

© 2008 by Taylor & Francis Group, LLC

we shall delve further into some of the fascinating complexities of quantization and compression of signals.

3.5 Quantization, entropy, and compression

In this section we shall examine some of the deeper questions involved with quantization and compression. One of the interesting features of quantizing wavelet transforms is the connection between structural properties of these transforms and basic ideas from information theory, such as entropy. This connection can be exploited to improve upon the basic compression results described in the preceding section.

Let's begin with a specific signal to be compressed, a recording of the author speaking the word *greasy.* A graph of the intensity values of this recording is shown in Figure 3.10(a). Notice that this graph consists of three main sections, summarized in Table 3.3. This signal exhibits within a short time-span a number of different linguistic effects, such as a quick transition from a low pitch *gr* sound to a higher pitch *e* sound, and then another quick transition to a very chaotically oscillating *s* sound, with a final transition back to a high pitch *y* sound. These effects make *greasy* a challenging test signal for any compression method.

Table 3.3 Main sections of *greasy.*

Section	*Time Interval*	*Sound*
1	$[0.00, 0.20]$	grea
2	$[0.25, 0.35]$	s
3	$[0.40, 0.60]$	y

The intensity values for this recording of *greasy* were quantized—using an 8-bit scalar quantization—as a part of the recording process. The sequences of bits encoding these 256 intensity values look like 01000001 or 11001000, and so on. A first bit of 1 indicates a negative intensity, while a first bit of 0 indicates a positive intensity. These bit sequences correspond to the integers $k = 0$ to $k = 255$, which we shall refer to as the *intensity levels.* To some extent this use of equal length bit sequences for all intensity levels is wasteful. This is indicated by the histogram of frequencies of occurrence of each intensity level shown in Figure 3.10(b). Since the most commonly occurring intensity level is the zero level, we can save bits if we encode this level with a single bit, such as 0.

By using shorter length bit sequences for the most commonly occurring intensity levels, and longer sequences for less commonly occurring intensity levels, we can reduce the total number of bits used. The idea is similar to Morse code where, for instance, the commonly occurring English letters *a* and *e* are encoded by the short sequences of dots and dashes $\cdot -$ and $\cdot$, respectively, while the less commonly occurring English letters *q* and *v* are encoded by

© 2008 by Taylor & Francis Group, LLC

the longer sequences $- - \cdot -$ and $\cdot \cdot \cdot -$, respectively. This procedure is made mathematically precise by fundamental results from the field known as *information theory.*

It is beyond the scope of this primer to provide a rigorous treatment of information theory. We shall just outline the basic ideas, and show how they apply to compressing *greasy.* Suppose that $\{p_k\}$ are the relative frequencies of occurrence of the intensity levels $k = 0$ through $k = 255$; that is, each p_k is an ordinate value in the histogram in Figure 3.10(b). Thus $p_k \geq 0$ for each k and $p_0 + p_1 + \cdots + p_{255} = 1$. These facts make it tempting to interpret each number p_k as a probability for the occurrence of k. Although these numbers p_k are not probabilities, nevertheless, a deterministic law governing the production of the intensity levels in *greasy* is *a priori* unknown to us. In fact, section 2 of *greasy,* as an isolated sound, is very similar to the random static background noise considered in the next section. There are deterministic models for producing sections 1 and 3, involving combinations of sinusoidal signals, but these are

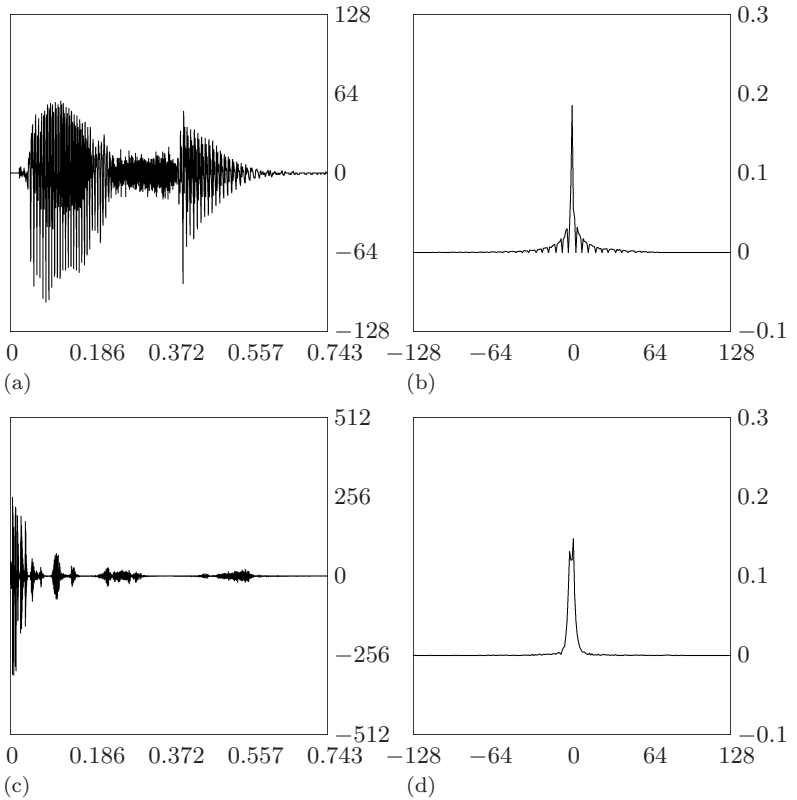


FIGURE 3.10
(a) The signal, *greasy.* (b) Histogram of 8-bit quantization of intensity levels of *greasy.* (c) 14-level Coif30 transform of *greasy.* (d) Histogram of 8-bit dead-zone quantization of the transform values.

© 2008 by Taylor & Francis Group, LLC

a posteriori models based on the recorded sound itself. In any case, let's see what consequences follow from treating the numbers p_k as probabilities for the occurrence of the intensity levels k. Let $\mathcal{L}_k$ be the length of the bit sequence that is used to encode the intensity level k. Then the *average length* $\overline{\mathcal{L}}$ of a lossless encoding of the k's is defined to be

$$\overline{\mathcal{L}} = p_0\mathcal{L}_0 + p_1\mathcal{L}_1 + \cdots + p_{255}\mathcal{L}_{255}\,. \tag{3.36}$$

The famous Shannon Coding Theorem tells us that $\overline{\mathcal{L}}$ satisfies the following inequality [if $p_k = 0$, then $p_k \log_2(1/p_k)$ is set equal to 0]:

$$\overline{\mathcal{L}} \;\geq\; p_0 \log_2 \frac{1}{p_0} + p_1 \log_2 \frac{1}{p_1} + \cdots + p_{255} \log_2 \frac{1}{p_{255}}\,. \tag{3.37}$$

The quantity on the right side of (3.37) is called the *entropy* for the probabilities p_k.

Inequality (3.37) says that the average length of any lossless encoding of the intensity levels k cannot be less than the entropy of the probabilities p_k in the histogram for these intensity levels. Or another, more accurate, way of putting things is that *a lossless encoding technique, such as Huffman coding or arithmetic coding, which is based on the relative frequencies p_k in the histogram for the k's, cannot achieve an average length less than the entropy.* For obvious reasons, these lossless coding techniques are called *entropy codings.*

For *greasy,* the entropy is found to be 5.43. Therefore, it is impossible to make an entropy coding of the intensity levels for *greasy* with any set of bit sequences whose average length is less than 5.43 bits. It should also be noted that, using either Huffman coding or arithmetic coding, it is possible to get within 1 bit or less of the entropy. For reasons of space, we shall not describe the methodologies of these encoding techniques; for another matter, these techniques are very well-known and there are many excellent descriptions of them to be found in the references for this chapter. Huffman codes are guaranteed, by the way in which they are constructed, to always achieve an average length that is within 1 bit of the entropy, while arithmetic codes can get asymptotically close to the entropy as the number of values to be encoded increases. Therefore, as a rough estimator of the average length of a well-chosen lossless encoding of the intensity levels we shall add 0.5 to the entropy. Using this estimator, we find that the 16,384 points of *greasy* can be expected to be entropy encoded with a total of $16,384 \times 5.93$ bits, i.e., using about 97,000 bits. This is not a particularly effective compression, since it still represents 5.93 bpp versus 8 bpp for the original signal.

The basis of wavelet transform encoding, as we explained in the last section, is to allow some inaccuracy resulting from quantizing transform values in order to achieve greater compression than by lossless methods. For instance, in Figure 3.10(c), we show a 14-level Coif30 transform of the *greasy* recording, and in Figure 3.10(d) we show a histogram of an 8-bit dead-zone quantization of this transform. Comparing the two histograms in Figure 3.10, we see that

© 2008 by Taylor & Francis Group, LLC

the histogram for the dead-zone quantization of the transform values is more narrowly concentrated than the histogram for the scalar quantization of the signal values. In fact, the entropy for the histogram in Figure 3.10 is 4.34, which is smaller than the entropy 5.43 for the histogram in Figure 3.10(c). Using our estimator for average coding length, we estimate that a lossless encoding of the quantized transform values will have an average length of 4.84. Consequently, the 3922 non-zero quantized transform values can be expected to be losslessly encoded with about $3922 \times 4.84 \approx 18,922$ bits. This is a significant reduction in the number of bits; even if absolutely no compression was done on the 16,384 bits in the significance map, there would still be a large improvement in compression over a lossless compression of *greasy.* It is very important to note that—even though there is some error introduced by the quantization of the transform values—when an inverse transform is performed, the resulting signal is extremely difficult to distinguish from the original when played over a computer sound system.

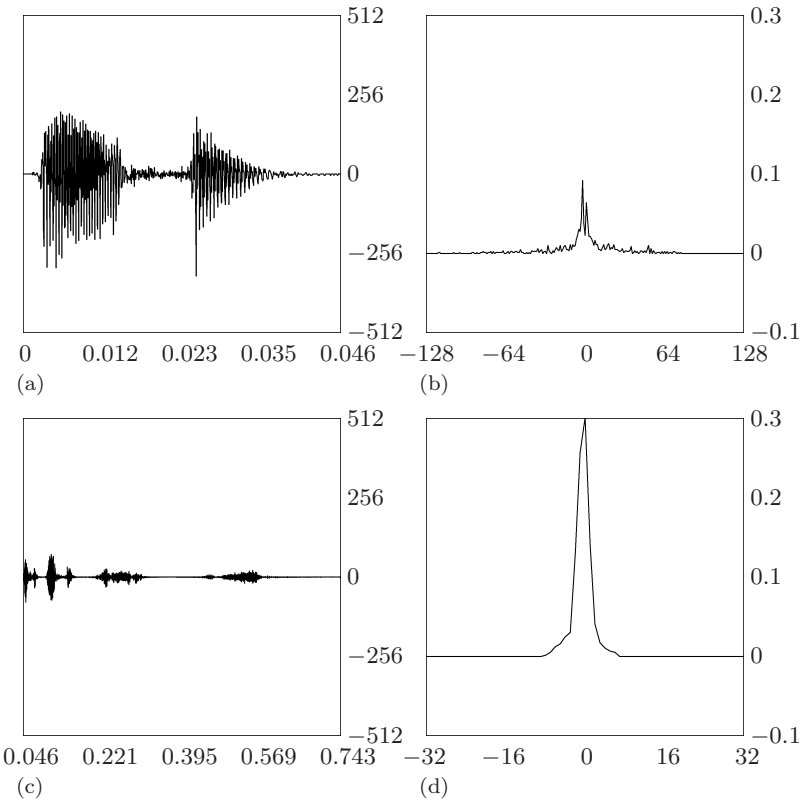


FIGURE 3.11
(a) Fourth trend of 4-level Coif30 transform of *greasy.* (b) Histogram of 8-bit dead-zone quantization of fourth trend. (c) Fluctuations of 4-level Coif30 transform. (d) Histogram of 6-bit dead-zone quantization of fluctuations.

© 2008 by Taylor & Francis Group, LLC

As another example of the possibilities available with wavelet transform compression, we shall examine a slightly different approach to compressing *greasy.* In Figures 3.11(a) and 3.11(c), we show a 4-level Coif30 transform of *greasy.* The fourth trend is graphed in Figure 3.11(a) and the 4 fluctuations are graphed in Figure 3.11(c). An 8-bit dead-zone quantization of the fourth trend has the histogram shown in Figure 3.11(b), with an entropy of 6.19. There are 793 non-zero coefficients; so encoding requires about $793 \times 6.69 \approx 5305$ bits. To quantize the fluctuation values we chose to use only 6 bits, not 8 bits as we used for the 14-level transform. A comparison of Figures 3.10(c) and 3.11(c) makes it apparent why we did this. More intensity levels are needed with the 14-level transform in order to ensure accuracy because of the many larger fluctuation values at higher levels [due to the higher peaks on the left side of Figure 3.10(c)]. This 6-bit dead-zone quantization of the four fluctuations has the histogram shown in Figure 3.11(d), with an entropy of 2.68. Since there are 2892 non-zero quantized values, an encoding requires about $2892 \times 3.18 \approx 9197$ bits. The total number of bits estimated for this 4-level transform compression is about $14,502$ bits. This compares favorably with the $18,982$ bits estimated for the 14-level transform compression. The improvement in compression for the 4-level transform is due to the decrease in entropy from 4.34 to 2.68 for the first four fluctuations brought about by the change made in the quantization.

This last example only begins to suggest some of the many possibilities available for adaptation of the basic wavelet transform compression procedure. For instance, one further possibility is to compute an entropy for a separate quantization of each fluctuation and separately encode these quantized fluctuations. This generally produces improvements in compression. For example, suppose a 4-level Coif30 transform of *greasy* is quantized using 8 bpp for the trend and 6 bpp for the four fluctuations, and separate entropies are calculated for the trend and for each of the four fluctuations. Then the estimated total number of bits needed is $13,107$. This is an improvement over the $14,502$ bits previously estimated.

3.6 Denoising audio signals

As we saw in Section 2.6, the problems of compression and noise removal are closely related. If a wavelet transform can effectively capture the energy of a signal in a few high-energy transform values, then additive noise can be effectively removed as well. We introduced a basic method, called *thresholding,* for removing noise, and illustrated this method using the Haar transform. Now, in this section, we shall threshold the Daubechies wavelet transforms. Besides discussing a simple illustrative example, we also shall give some justification for why thresholding works well for the random noise often encountered in signal transmission, and provide an example of denoising when the noise is a combination of pop noise and random noise.

Let's quickly review the basic steps for removing additive noise using the

© 2008 by Taylor & Francis Group, LLC

Threshold Method. A threshold value T is chosen for which all transform values that are lesser in magnitude than T are set equal to zero. By performing an inverse wavelet transform on the thresholded transform, an estimation of the original uncontaminated signal is obtained. In Chapter 2 we saw that the Haar transform was able to effectively remove the noise from Signal A, as shown in Figure 2.6 on p. 28. Signal A was created by adding random noise to Signal 1, whose graph is shown in Figure 2.4 on p. 23. Because of the connection between compression and noise removal, and because the Haar transform is the most effective transform for compressing Signal 1, it follows that the Haar transform is also the most effective transform for denoising Signal A.

With the other noisy signal examined in Section 2.6, Signal B, we found that the Haar transform did a rather poor job of denoising. The explanation for this lies again in the connection between compression and threshold denoising. Signal B was created by adding random noise to Signal 2, but the Haar transform is not an effective tool for compressing Signal 2; it produces too many low magnitude transform values which are obscured by the noise. We saw, however, in Section 3.4 that the Coif30 transform is a very effective tool for compressing Signal 2. Therefore, let's apply it to denoising Signal B.

In Figure 3.12 we show the basic steps in a threshold denoising of Signal B using a 12-level Coif30 transform. Comparing the 12-level Coif30 transforms of Signal B and Signal 2, we see that the addition of noise has contributed a large number of small magnitude, low-energy values to the transform of Signal 2. Nevertheless, most of the high-energy values of the transform of Signal 2 are still plainly visible in the transform of Signal B, although their values have been altered slightly by the added noise. Therefore, we can eliminate most of the noise using thresholding, as indicated by the two horizontal lines in Figure 3.12(b). All transform values between ± 0.2 are set equal to zero, and this produces the thresholded transform shown in Figure 3.12(c). Comparing this thresholded transform with the transform of Signal 2, shown in Figure 3.8(b) on p. 62, we can see that thresholding has produced a fairly close match. The most noticeable difference is a loss of a small number of values located near 0.25, which had such small magnitudes that they were obscured by the addition of the noise. Since the two transforms are such a reasonably close match, it follows that the inverse transform of the thresholded transform produces a denoised signal that is a close match of Signal 2. In fact, the RMS Error between the denoised signal and Signal 2 is 0.14, which is a four-fold decrease in the RMS Error of 0.57 between Signal B and Signal 2.

3.6.1 Choosing a threshold value

One of the most attractive features of wavelet threshold denoising is that, for the type of random noise frequently encountered in signal transmission, it is possible to automatically choose a threshold for denoising *without any prior knowledge of the signal.* In Figure 3.13(a) we show an example of this type of

© 2008 by Taylor & Francis Group, LLC

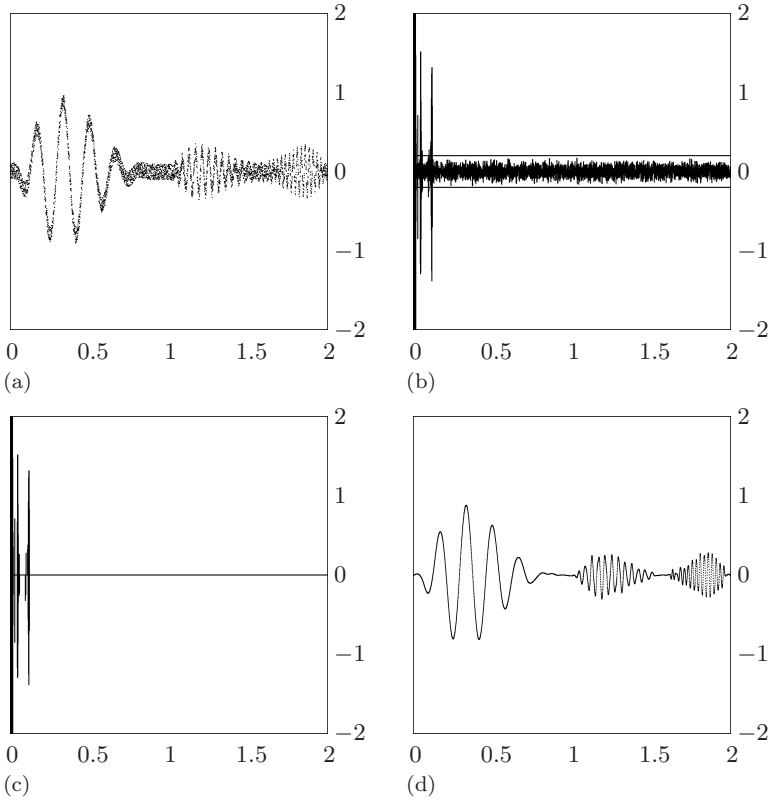


FIGURE 3.12
(a) Signal B. (b) 12-level Coif30 transform, with thresholding indicated by two horizontal lines at ± 0.2. (c) Thresholded transform. (d) Denoised signal; compare with Figure 3.8(a) on p. 62 and Figure 2.7(d) on p. 30.

random noise; it is called *Gaussian noise*. A complete discussion of Gaussian random noise is beyond the scope of this book; in this subsection we shall only give a brief introduction to some of the main ideas.

If the noise shown in Figure 3.13(a) is played over a sound generator, it produces the familiar static sound that can be heard in noisy radio transmissions and analog recordings. A histogram of frequencies of occurrence of intensity levels for this noise is shown in Figure 3.13(b). The bell-shaped curve that this histogram approximates is an indication that this noise is Gaussian. Using elementary statistical formulas on the noise values, we can estimate the mean μ and standard deviation σ of the probability density function (the bell-shaped curve) that this histogram approximates. We find that the mean is approximately 0 and that the standard deviation is approximately 0.505. One of the consequences of the conservation of energy property of the Daubechies wavelet transforms—more precisely, the orthogonality of the matrix form for

© 2008 by Taylor & Francis Group, LLC

these transforms[7]—is that they preserve the Gaussian nature of the noise. For example, in Figure 3.13(c) we show a Coif30 wavelet transform of the random noise in Figure 3.13(a). Its histogram in Figure 3.13(d) approximates a bell-shaped curve, and the mean and standard deviation of the transformed noise are calculated to be approximately 0 and 0.505, the same values that we found for the original noise.

These facts imply that the transformed noise values will be similar in magnitude to the original noise values and, more importantly, that *a large percentage of these values will be smaller in magnitude than a threshold equal to a large enough multiple of the standard deviation* σ. To see this last point more clearly, consider the data shown in Table 3.4. In this table, the first

Table 3.4 Comparison of three noise histograms

Threshold	*% below, Theory*	*% below, 3.13(a)*	*% below, 3.13(c)*
σ	68.72	68.87	69.94
2σ	95.45	95.70	95.39
3σ	99.73	99.74	99.66
4σ	99.99	100.00	99.98

column contains four thresholds which are multiples of the standard deviation $\sigma = 0.505$. The second column lists the percentages of values of random numbers having magnitudes that are less than these thresholds, assuming that these random numbers obey a Gaussian normal probability law with mean 0 and standard deviation σ.[8] The third and fourth columns contain the percentages of the magnitudes from the noise signals in Figure 3.13(a) and 3.13(c) which lie below the thresholds.

Based on the results of this table, we shall use the following formula:

$$T = 4.5\sigma \tag{3.38}$$

for setting the threshold value T. In our next example, we shall make use of (3.38) for choosing a threshold. The standard deviation σ can be estimated from a portion of the transform which consists largely of noise values. Generally this is the case with the first level fluctuation because the first level fluctuation values from the original signal are typically very small.[9] When Formula (3.38) is used we expect that well over 99% of the noise values will be removed from the transform, and this requires essentially no knowledge of the original, uncontaminated signal. Preventing this thresholding from removing too many transform values from the original signal, however, depends

[7]Orthogonality of the Daubechies wavelet transforms was discussed in Section 3.2.

[8]That is, the probability of a number x having magnitude less than T is equal to the area under the curve $\frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2}$ from $-T$ to T.

[9]See the subsection, **Estimation of noise standard deviation,** on p. 136 for details on how σ is estimated from first level fluctuation values.

© 2008 by Taylor & Francis Group, LLC

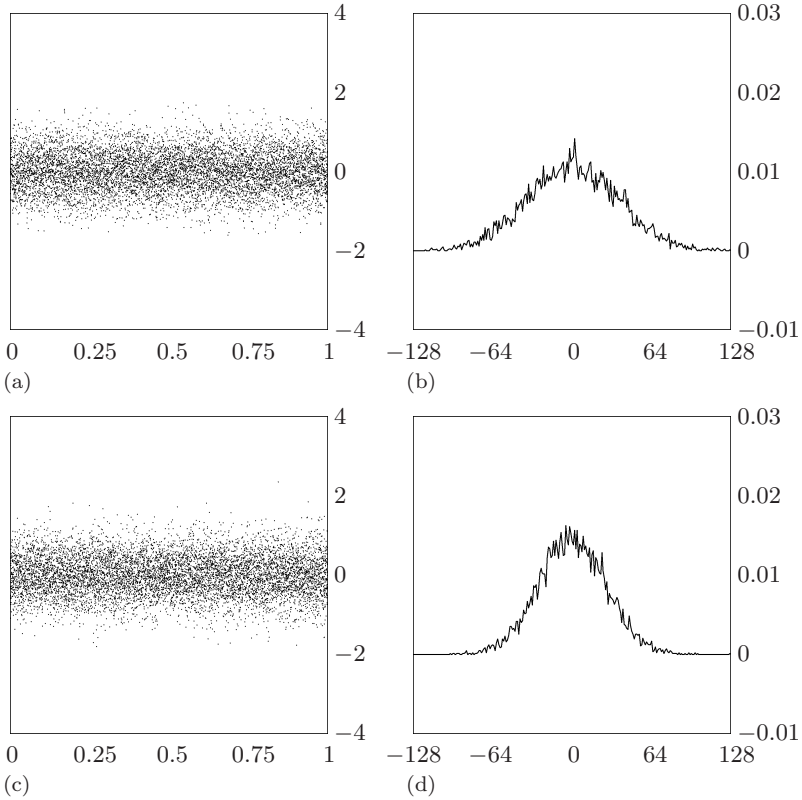


FIGURE 3.13
(a) Gaussian noise. (b) Histogram of the noise. (c) Coif30 transform of the noise. (d) Histogram of the transformed noise.

on how well the transform compresses the energy of the signal into a few high-magnitude values which stand out from the threshold. With the Daubechies transforms this will occur with signals that are sampled from analog signals that are smooth. This is because so many of the fluctuation values are small, and that leaves only a few high-magnitude transform values to account for the energy. Of course it is always possible—by adding noise with a sufficiently high standard deviation—to produce such a high threshold that the signal's transform values are completely wiped out by thresholding. Wavelet thresholding is powerful, but it cannot perform magic.

3.6.2 Removing pop noise and background static

We close this section by discussing another example of noise removal. In Figure 3.14(a) we show the graph of a noisy signal. The signal consists of Gaussian random noise and pop noise added to a recording of the author making a whistling sound. The pop noise consists of the spike located near

© 2008 by Taylor & Francis Group, LLC

0.186, which is plainly visible in the graph. When this signal is played over a computer sound system, the whistle can be heard but there is an annoying static background and a rather loud, short, popping sound near the beginning of the whistle. Using wavelet techniques we can remove essentially all of the random noise and greatly reduce the volume of the pop noise.

To remove the noise, we use a Coif18 wavelet transform of the signal. In Figure 3.14(b) we show the second fluctuation subsignal of a Coif18 transform of the noisy signal. The other fluctuation subsignals are similar in appearance; so by describing how to modify this particular fluctuation we shall be describing how to handle the other fluctuations as well. The random noise, or background static, appears as random oscillations in the wavelet transform as we noted above. In fact, on the interval $[0.46, 0.51]$ the fluctuation values appear purely random; so we make an estimate of the standard deviation of the transform noise over this subinterval, obtaining $\sigma \approx 0.66086$. By Formula (3.38), we should set T equal to 2.97387. Rounding up slightly, we set $T = 2.98$.

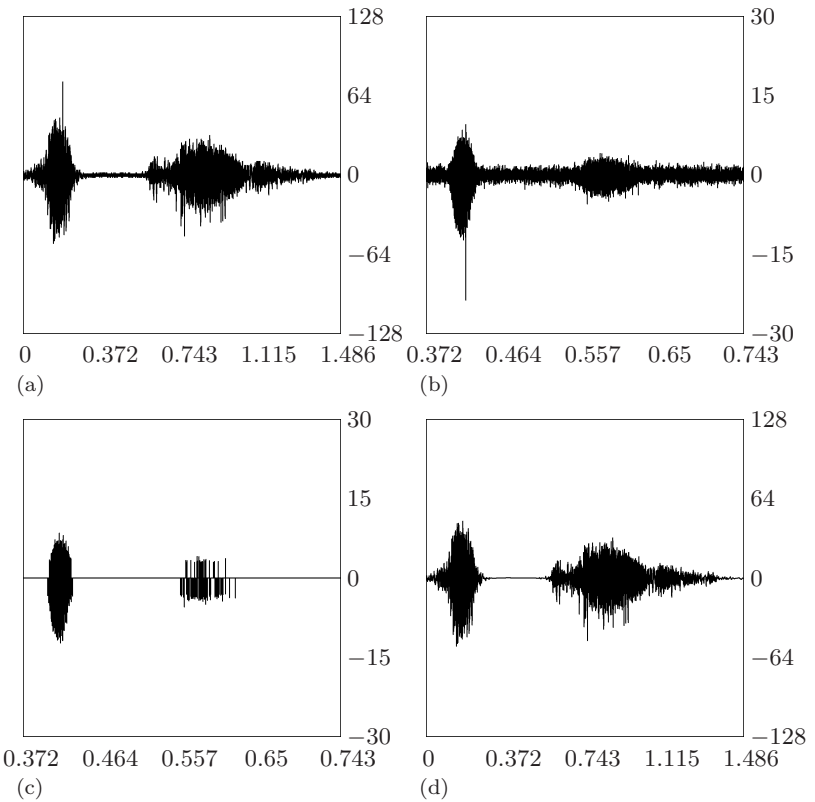


FIGURE 3.14
(a) Noisy whistle. (b) Second fluctuation of its 15-level Coif18 transform. (c) Second fluctuation of denoised transform. (d) Denoised whistle.

© 2008 by Taylor & Francis Group, LLC

In contrast to the random noise, the pop noise produces unusually large magnitude fluctuation values that stand out from the vast majority of the fluctuation values. We shall refer to these unusually large fluctuation values as *outliers.* For example, in Figure 3.14(b) there is an outlier clearly visible in the second fluctuation. It appears as a spike on the left side of the figure. To remove the pop noise we must eliminate such outliers from the transform.

Table 3.5 Acceptance bands

Fluctuation	*Acceptance band*
1	$(-4, -T) \cup [T, 4)$
2	$(-12.5, -T) \cup (T, 9)$
3	$(-\infty, -T) \cup (T, \infty)$
4	$(-67, -T) \cup (T, 67)$
5 to 15	$(-\infty, -T) \cup (T, \infty)$

To remove both the random noise values and the outliers from the transform, we set *acceptance bands* for each fluctuation subsignal. Values that lie in the acceptance band for each fluctuation are retained and all other values are set equal to zero. These acceptance bands, which were obtained by a visual inspection of the transform, are summarized in Table 3.5. The second fluctuation of the modified, denoised transform is shown in Figure 3.14(c). By performing an inverse transform on the denoised transform, we produce the denoised whistle signal shown in Figure 3.14(d). When this denoised signal is played on a computer's sound system, the background static is completely gone and the volume of the pop noise is so greatly diminished that it is barely audible.

No claim is being made here that the method used for removing the pop noise is in any way optimal. There are two problems with the approach used. First, we only removed outliers from the fluctuation levels 1, 2, and 4. An examination of the transform reveals outliers for several other levels, such as 5 and 6. A more effective denoising would remove these spikes as well. Furthermore, actually removing the spikes completely may not be the best approach; reducing their size based on an estimate of nearby fluctuation values might work better. In any case, this example was described in order to indicate some of the flexibility available in wavelet denoising. More detailed discussions of denoising methods can be found in the references.

3.7 Biorthogonal wavelets

In this section and the next, we provide a brief account of *biorthogonal wavelets.* These wavelets are a useful generalization of the orthogonal wavelets that we have considered up to now. They are particularly important in the image processing applications that we discuss in the next chapter. We shall

© 2008 by Taylor & Francis Group, LLC

examine two of the most important biorthogonal systems: the Daub 5/3 system which plays a role in lossless image compression, and the Daub 9/7 system which is used for both lossy image compression and denoising. We shall also discuss the Daub 5/3 integer-to-integer transform which is a crucial component of lossless image compression.

In this section, we focus primarily on the Daub 5/3 system, which provides an elementary model for all biorthogonal systems. At the end, we show how the Daub 5/3 wavelet transform can be converted into a transform that maps integers to integers. In the next section, we describe the Daub 9/7 system as a generalization of the Daub 5/3 system.

3.7.1 Daub 5/3 system

The Daub 5/3 biorthogonal system is the simplest biorthogonal system. It is called *biorthogonal* for two reasons: (1) the Daub 5/3 wavelet transform is *not* energy preserving (it is not an orthogonal transform); (2) one set of basis signals is used for calculating transform values while a second set of basis signals is used for the MRA expansion of a signal (these two bases are related by *biorthogonality conditions* which we define later).

We shall now describe the Daub 5/3 wavelet transform. The 1-level Daub 5/3 transform is the mapping $\mathbf{f} \overset{\mathbf{D}_1}{\longmapsto} (\mathbf{a}^1 \,|\, \mathbf{d}^1)$ where the trend subsignal $\mathbf{a}^1$ is calculated using the formula

$$a_k = \frac{-1}{8} f_{2k-3} + \frac{1}{4} f_{2k-2} + \frac{3}{4} f_{2k-1} + \frac{1}{4} f_{2k} + \frac{-1}{8} f_{2k+1} \tag{3.39}$$

and the fluctuation subsignal $\mathbf{d}^1$ is calculated using the formula

$$d_k = \frac{-1}{2} f_{2k-1} + f_{2k} + \frac{-1}{2} f_{2k+1}. \tag{3.40}$$

Notice the symmetry of the coefficients in these two formulas; this will lead to symmetric scaling signals and wavelets.

There is one wrinkle here. Although Formula (3.39) can be used as it stands for $k = 2, \ldots, N/2 - 1$, we need to extend the values of the signal $\mathbf{f}$ beyond its endpoint indices 1 and N in order to use (3.39) for $k = 1$ and $k = N/2$. Up till now, we made such extensions by periodicity (wrap-around). But the symmetry of the coefficients in Formula (3.39) practically demands that we make a symmetric extension of $\mathbf{f}$. We extend $\mathbf{f}$ by *even symmetry* at its endpoints by defining $f_0 = f_2$, $f_{-1} = f_3$, and $f_{N+1} = f_{N-1}$, $f_{N+2} = f_{N-2}$, and so on. With this extension of $\mathbf{f}$, both (3.39) and (3.40) can be applied for $k = 1, \ldots, N/2$ to generate the values of $\mathbf{a}^1$ and $\mathbf{d}^1$.

Based on Equation (3.39), with symmetric extension for $k = 1$ and $k = N$,

© 2008 by Taylor & Francis Group, LLC

we define the Daub 5/3 *analysis scaling signals* $\{\mathbf{V}_k^1\}$ as follows:

$$\begin{aligned}
\mathbf{V}_1^1 &= (\frac{3}{4}, \frac{1}{2}, \frac{-1}{4}, 0, \ldots, 0) \\
\mathbf{V}_2^1 &= (\frac{-1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{-1}{8}, 0, \ldots, 0) \\
\mathbf{V}_3^1 &= (0, 0, \frac{-1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{-1}{8}, 0, \ldots, 0) \\
&\vdots \\
\mathbf{V}_{N/2}^1 &= (0, \ldots, 0, \frac{-1}{8}, \frac{1}{4}, \frac{5}{8}, \frac{1}{4}).
\end{aligned}$$

Using these analysis scaling vectors, we express Equation (3.39) as $a_k = \mathbf{f} \cdot \mathbf{V}_k^1$. In fact, however, *since we use even symmetric extension,* we can express these analysis scaling vectors in a more general way. Let the *analysis scaling numbers* be defined as

$$\alpha_1 = \frac{-1}{8}\,,\ \alpha_2 = \frac{1}{4}\,,\ \alpha_3 = \frac{3}{4}\,,\ \alpha_4 = \frac{1}{4}\,,\ \alpha_5 = \frac{-1}{8}.$$

Then, the general form of an analysis scaling vector is

$$\mathbf{V}_k^1 = (0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, 0, \ldots, 0)$$

and $\mathbf{V}_{k+1}^1$ is a translation of $\mathbf{V}_k^1$ by two time-units, and $a_k = \mathbf{f} \cdot \mathbf{V}_k^1$ for each k.

Notice that these analysis scaling numbers satisfy

$$\begin{aligned}
\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 &= 1 \\
(-2)\,\alpha_1 + (-1)\,\alpha_2 + 0\,\alpha_3 + 1\,\alpha_4 + 2\,\alpha_5 &= 0.
\end{aligned}$$

Analogous to the coiflets, these last two equations imply that the trend values at any given level are often close matches of an analog signal ($\mathbf{a}_m^1 \approx g(t_{2m})$, $\mathbf{a}_m^2 \approx g(t_{4m})$, and so on[10]).

In a similar way, we define the *analysis wavelets* $\{\mathbf{W}_k^1\}$ by the equations

$$\begin{aligned}
\mathbf{W}_1^1 &= (\frac{-1}{2}, 1, \frac{-1}{2}, 0, 0, \ldots, 0) \\
\mathbf{W}_2^1 &= (0, 0, \frac{-1}{2}, 1, \frac{-1}{2}, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{W}_{N/2}^1 &= (0, \ldots, 0, -1, 1)
\end{aligned}$$

[10]No multiplication by $\sqrt{2}$ occurs with the Daub 5/3 system because the sum of the analysis scaling numbers is 1 rather than $\sqrt{2}$.

© 2008 by Taylor & Francis Group, LLC

and Equation (3.40) can then be expressed as $d_k = \mathbf{f} \cdot \mathbf{W}_k^1$. By defining the analysis wavelet numbers as

$$\beta_1 = \frac{-1}{2}\,,\ \beta_2 = 1\,,\ \beta_3 = \frac{-1}{2}$$

the general form of an *analysis wavelet* is

$$\mathbf{W}_k^1 = (0, \ldots, 0, \beta_1, \beta_2, \beta_3, 0, \ldots, 0)$$

with $\mathbf{W}_{k+1}^1$ being a translation of $\mathbf{W}_k^1$ by two time-units, and $d_k = \mathbf{f} \cdot \mathbf{W}_k^1$ for each k. The Daub 5/3 wavelet transform produces many small fluctuation values because the analysis wavelet numbers satisfy

$$\begin{aligned} \beta_1 + \beta_2 + \beta_3 &= 0 \\ (-1)\,\beta_1 + 0\,\beta_2 + 1\,\beta_3 &= 0. \end{aligned}$$

These equations are analogous to the ones satisfied by the Daub4 wavelet numbers and they typically produce many small fluctuation values for the same reasons as we discussed for the Daub4 wavelet transform.

3.7.2 Daub 5/3 inverse

The 1-level Daub 5/3 transform has an inverse. The simplest way to define it is to note that Equations (3.39) and (3.40) can be rewritten as

$$d_k = f_{2k} - \frac{1}{2}(f_{2k-1} + f_{2k+1}) \tag{3.41a}$$

$$a_k = f_{2k-1} + \frac{1}{4}(d_{k-1} + d_k) \tag{3.41b}$$

where all values of $\{d_k\}$ are computed first using (3.41a) and then used in the right side of (3.41b) to compute the values a_k. These last two equations are called the *lifting form* of the Daub 5/3 transform. It is easy to see how to invert them. Their inverse is given by the equations

$$f_{2k-1} = a_k - \frac{1}{4}(d_{k-1} + d_k) \tag{3.42a}$$

$$f_{2k} = d_k + \frac{1}{2}(f_{2k-1} + f_{2k+1}). \tag{3.42b}$$

3.7.3 MRA for the Daub 5/3 system

As with all our previous wavelet systems, the inverse Daub 5/3 transform

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) \overset{\mathbf{D}_1^{-1}}{\longmapsto} \mathbf{f}$$

can be expressed as the MRA equation $\mathbf{f} = \mathbf{A}^1 + \mathbf{D}^1$. We could develop the expansions of $\mathbf{A}^1$ and $\mathbf{D}^1$ into combinations of scaling signals and wavelets

© 2008 by Taylor & Francis Group, LLC

from the defining equations of the inverse of the Daub 5/3 transform, like we did with the Haar transform, but there is a more elegant way that makes use of the linearity of the Daub 5/3 transform.

The Daub 5/3 transform is *linear* in the sense that it satisfies the following identity for all signals $\mathbf{f}$ and $\mathbf{g}$ and constants r and s:

$$r\mathbf{f} + s\mathbf{g} \overset{\mathbf{D}_1}{\longmapsto} r\mathbf{D}_1(\mathbf{f}) + s\mathbf{D}_1(\mathbf{g}).$$

Applying the Daub 5/3 inverse $\mathbf{D}_1^{-1}$ to both sides of this mapping, we obtain

$$r\mathbf{D}_1(\mathbf{f}) + s\mathbf{D}_1(\mathbf{g}) \overset{\mathbf{D}_1^{-1}}{\longmapsto} r\mathbf{f} + s\mathbf{g}$$

for all signals $\mathbf{f}$ and $\mathbf{g}$ and constants r and s, which shows that the inverse transform is also linear. Applying linearity of the inverse systematically to

$$\begin{aligned}\mathbf{D}_1(\mathbf{f}) &= (a_1, a_2, \ldots, a_{N/2}, \,|\, d_1, d_2, \ldots, d_{N/2}) \\ &= a_1(1,0,\ldots,0\,|\,0,\ldots,0) + \cdots + a_{N/2}(0,\ldots,0,1\,|\,0,\ldots,0) \\ &\quad + d_1(0,\ldots,0\,|\,1,0,\ldots,0) + \cdots + d_{N/2}(0,\ldots,0\,|\,0,\ldots,0,1)\end{aligned}$$

we obtain

$$\begin{aligned}\mathbf{f} = {} & a_1\mathbf{D}_1^{-1}(1,0,\ldots,0\,|\,0,\ldots,0) + \cdots + a_{N/2}\mathbf{D}_1^{-1}(0,\ldots,0,1\,|\,0,\ldots,0) \\ & + d_1\mathbf{D}_1^{-1}(0,\ldots,0\,|\,1,0,\ldots,0) + \cdots + d_{N/2}\mathbf{D}_1^{-1}(0,\ldots,0\,|\,0,\ldots,0,1).\end{aligned}$$

Based on this last equation, we define the Daub 5/3 *synthesis scaling signals* $\{\widetilde{\mathbf{V}}_k^1\}$ and Daub 5/3 *synthesis wavelets* $\{\widetilde{\mathbf{W}}_k^1\}$ as

$$\begin{aligned}\widetilde{\mathbf{V}}_1^1 &= \mathbf{D}_1^{-1}(1,0,\ldots,0\,|\,0,\ldots,0) \\ &\vdots \\ \widetilde{\mathbf{V}}_{N/2}^1 &= \mathbf{D}_1^{-1}(0,\ldots,0,1\,|\,0,\ldots,0) \\ \widetilde{\mathbf{W}}_1^1 &= \mathbf{D}_1^{-1}(0,\ldots,0\,|\,1,0,\ldots,0) \\ &\vdots \\ \widetilde{\mathbf{W}}_{N/2}^1 &= \mathbf{D}_1^{-1}(0,\ldots,0\,|\,0,\ldots,0,1).\end{aligned} \tag{3.43}$$

Thereby obtaining the following MRA of $\mathbf{f}$:

$$\begin{aligned}\mathbf{f} &= \mathbf{A}^1 + \mathbf{D}^1 \\ &= a_1\widetilde{\mathbf{V}}_1^1 + \cdots + a_k\widetilde{\mathbf{V}}_k^1 + \cdots + a_{N/2}\widetilde{\mathbf{V}}_{N/2}^1 \\ &\quad + d_1\widetilde{\mathbf{W}}_1^1 + \cdots + d_k\widetilde{\mathbf{W}}_k^1 + \cdots + d_{N/2}\widetilde{\mathbf{W}}_{N/2}^1\end{aligned}$$

with

$$a_k = \mathbf{f}\cdot\mathbf{V}_k^1, \quad d_k = \mathbf{f}\cdot\mathbf{W}_k^1.$$

© 2008 by Taylor & Francis Group, LLC

Notice that these last equations imply that

$$\widetilde{\mathbf{V}}_j^1 \cdot \mathbf{V}_k^1 = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k, \end{cases} \qquad \widetilde{\mathbf{W}}_j^1 \cdot \mathbf{V}_k^1 = 0, \text{ all } j, k,$$

$$\widetilde{\mathbf{W}}_j^1 \cdot \mathbf{W}_k^1 = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k, \end{cases} \qquad \widetilde{\mathbf{V}}_j^1 \cdot \mathbf{W}_k^1 = 0, \text{ all } j, k,$$

which are known as the *biorthogonality relations.* All biorthogonal systems must satisfy them.

For the Daub 5/3 system, the formulas in (3.43) yield (using symmetric extension) the following synthesis signals:

$$\begin{aligned}
\widetilde{\mathbf{V}}_1^1 &= (1, \frac{1}{2}, 0, \dots, 0) \\
\widetilde{\mathbf{V}}_2^1 &= (0, \frac{1}{2}, 1, \frac{1}{2}, 0, \dots, 0) \\
\widetilde{\mathbf{V}}_3^1 &= (0, 0, 0, \frac{1}{2}, 1, \frac{1}{2}, 0, \dots, 0) \\
&\vdots \\
\widetilde{\mathbf{V}}_{N/2}^1 &= (0, \dots, 0, \frac{1}{2}, 1, 1) \\
\widetilde{\mathbf{W}}_1^1 &= (\frac{-1}{2}, \frac{5}{8}, \frac{-1}{4}, \frac{-1}{8}, 0, \dots, 0) \\
\widetilde{\mathbf{W}}_2^1 &= (0, \frac{-1}{8}, \frac{-1}{4}, \frac{3}{4}, \frac{-1}{4}, \frac{-1}{8}, 0, \dots, 0) \\
\widetilde{\mathbf{W}}_3^1 &= (0, 0, 0, \frac{-1}{8}, \frac{-1}{4}, \frac{3}{4}, \frac{-1}{4}, \frac{-1}{8}, 0, \dots, 0) \\
&\vdots \\
\widetilde{\mathbf{W}}_{N/2}^1 &= (0, \dots, 0, \frac{-1}{8}, \frac{-1}{4}, \frac{3}{4}).
\end{aligned}$$

3.7.4 Daub 5/3 transform, multiple levels

As usual with wavelet transforms, we extend the Daub 5/3 transform to higher levels by iterating it on preceding trend subsignals (with even symmetric extension as needed). For example, the 2-level Daub 5/3 transform $\mathbf{f} \overset{\mathbf{D}_2}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1)$ is obtained by applying Equations (3.39) and (3.40)—or even better, Equations (3.41a) and (3.41b)—with the values of $\mathbf{a}^1$ substituted for those of $\mathbf{f}$ on the right sides (using even symmetry as needed). The MRA

$$\mathbf{f} = \mathbf{A}^m + \mathbf{D}^m + \mathbf{D}^{m-1} + \cdots + \mathbf{D}^1$$

is constructed by applying the inverse of the m-level Daub 5/3 transform:

$$(\mathbf{a}^m \,|\, \mathbf{d}^m \,|\, \mathbf{d}^{m-1} \,|\, \dots \,|\, \mathbf{d}^1) \overset{\mathbf{D}_m^{-1}}{\longmapsto} \mathbf{A}^m + \mathbf{D}^m + \mathbf{D}^{m-1} + \cdots + \mathbf{D}^1 = \mathbf{f}.$$

© 2008 by Taylor & Francis Group, LLC

At every level m, the averaged signal $\mathbf{A}^m$ is a sum of multiples of synthesis scaling signals $\{\widetilde{\mathbf{V}}_k^m\}$. Each $\widetilde{\mathbf{V}}_k^m$ is obtained by applying the inverse of the m-level Daub 5/3 transform to a standard basis vector:

$$\widetilde{\mathbf{V}}_k^m = \mathbf{D}_m^{-1}(0, \ldots, 0, 1, 0, \ldots, 0)$$

where the 1 is in the k^{th} position, $k = 1, \ldots, N/2^m$. Likewise, the detail signal $\mathbf{D}^m$ is a sum of multiples of synthesis wavelets $\{\widetilde{\mathbf{W}}_k^m\}$. Each $\widetilde{\mathbf{W}}_k^m$ is obtained by applying the inverse of the m-level Daub 5/3 transform to a standard basis vector:

$$\widetilde{\mathbf{W}}_k^m = \mathbf{D}_m^{-1}(0, \ldots, 0, 1, 0, \ldots, 0)$$

where the 1 is in the $k + N/2^m$ position, $k = 1, \ldots, N/2^m$.

The symmetry of these synthesis signals $\widetilde{\mathbf{V}}_k^m$ and $\widetilde{\mathbf{W}}_k^m$ is illustrated in Figures 3.15(c) and 3.15(d). The symmetry of the scaling and wavelet numbers

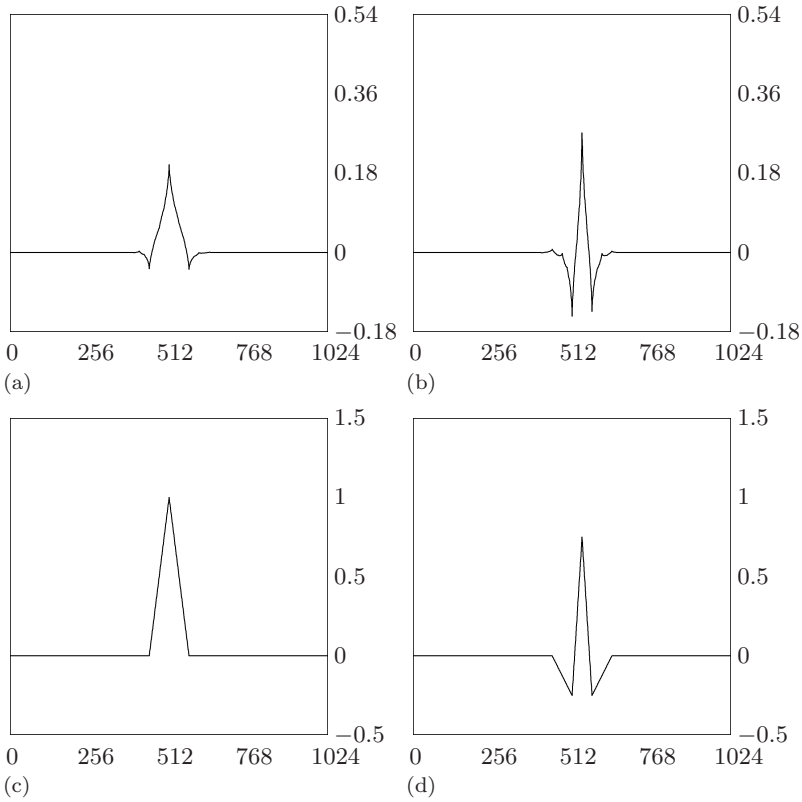


FIGURE 3.15
Graphs of scaling functions and wavelets using 1024 points. (a) Graph of Coif6 scaling function $\mathbf{V}_9^6$. (b) Graph of Coif6 wavelet $\mathbf{W}_9^6$. (c) Graph of Daub 5/3 synthesis scaling function $\widetilde{\mathbf{V}}_9^6$. (d) Graph of Daub 5/3 synthesis wavelet $\widetilde{\mathbf{W}}_9^6$. The Daub 5/3 signals are symmetric, but the Coif6 signals are not.

© 2008 by Taylor & Francis Group, LLC

for the Daub 5/3 transform leads to symmetry of these scaling signals and wavelets.

Unlike the other wavelet transforms we have considered, the Daub 5/3 transform does not preserve energy. For example,

$$(1,0,\ldots,0) \xrightarrow{\mathbf{D}_1} (\frac{3}{4},\frac{-1}{8},0,\ldots,0\,|\,\frac{-1}{2},0,\ldots,0)$$

so a signal of energy 1 is transformed to a signal of energy 53/64. We shall see in the next chapter how to slightly modify the Daub 5/3 transform of images so that it is nearly energy preserving and also retains all of the other features (symmetry, small fluctuations, etc.) of the original Daub 5/3 system.

3.7.5 Daub 5/3 integer-to-integer system

The Daub 5/3 transform can be recast into a new form that maps integer-valued signals to integer-valued signals. This new transform, the Daub 5/3 integer-to-integer transform, is used in image processing to produce lossless compressions of images.

The Daub 5/3 integer-to-integer transform is accomplished by the following modification of Equations (3.41a) and (3.41b):

$$d_k = f_{2k} - \left\lfloor \frac{1}{2}(f_{2k-1} + f_{2k+1}) + \frac{1}{2} \right\rfloor \tag{3.44a}$$

$$a_k = f_{2k-1} + \left\lfloor \frac{1}{4}(d_{k-1} + d_k) + \frac{1}{2} \right\rfloor \tag{3.44b}$$

where $\lfloor x+1/2 \rfloor$ rounds x to a nearest integer. Whenever the signal $\mathbf{f}$ has integer values these equations are guaranteed to produce integer-valued transform values. They define the 1-level *Daub 5/3 integer-to-integer transform.* All values of $\{d_k\}$ are computed first via (3.44a) and then all values of $\{a_k\}$ are computed via (3.44b). As discussed above, whenever necessary, even symmetry is imposed on our signals when applying Equations (3.44a) and (3.44b).

The inverses of Equations (3.44a) and (3.44b) are

$$f_{2k-1} = a_k - \left\lfloor \frac{1}{4}(d_{k-1} + d_k) + \frac{1}{2} \right\rfloor \tag{3.45a}$$

$$f_{2k} = d_k + \left\lfloor \frac{1}{2}(f_{2k-1} + f_{2k+1}) + \frac{1}{2} \right\rfloor, \tag{3.45b}$$

where all odd-indexed values $\{f_{2k-1}\}$ are computed first using (3.45a) and then the even-indexed values $\{f_{2k}\}$ are computed using (3.45b).

Higher level Daub 5/3 integer-to-integer transforms are calculated by applying Equations (3.44a) and (3.44b) recursively on trend values. For example, $\mathbf{a}^2$ is computed using the values of $\mathbf{a}^1$ on the right sides of (3.44a) and (3.44b). Equations (3.45a) and (3.45b) are used recursively to compute inverses of these higher level transforms.

© 2008 by Taylor & Francis Group, LLC

3.8 The Daub 9/7 system

The Daub 9/7 system is an important generalization of the Daub 5/3 system. It plays a crucial role in lossy image compression and denoising. It has proven to be so successful at image compression that it has been adopted by the new image compression standard, JPEG 2000, which we will examine in the next chapter.

The Daub 9/7 system is defined in a very similar way as the Daub 5/3 system, so we shall only discuss it briefly. More details can be found in the papers of Wim Sweldens and his collaborators listed in the Notes and References section for this chapter.

The Daub 9/7 analysis signals are defined using the following numbers:

$$\begin{aligned}
\alpha_1 &= 0.0378284554956993 & \beta_1 &= 0.0645388826835489 \\
\alpha_2 &= -0.0238494650131592 & \beta_2 &= -0.0406894176455255 \\
\alpha_3 &= -0.110624404085811 & \beta_3 &= -0.418092272881996 \\
\alpha_4 &= 0.377402855512633 & \beta_4 &= 0.788485616984644 \\
\alpha_5 &= 0.852698678836979 & \beta_5 &= -0.418092272881996 \\
\alpha_6 &= 0.377402855512633 & \beta_6 &= -0.0406894176455255 \\
\alpha_7 &= -0.110624404085811 & \beta_7 &= 0.0645388826835489 \\
\alpha_8 &= -0.0238494650131592 & & \\
\alpha_9 &= 0.0378284554956993 & . &
\end{aligned}$$

Using these numbers, the typical analysis scaling signal is

$$\mathbf{V}_k^1 = (0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9, 0, \ldots, 0)$$

with $\mathbf{V}_{k+1}^1$ a translation by two time-units of $\mathbf{V}_k^1$ and each trend value computed as $a_k = \mathbf{f} \cdot \mathbf{V}_k^1$. As with the Daub 5/3 system, we assume even symmetric extension of our signals. Likewise, the typical analysis wavelet is

$$\mathbf{W}_k^1 = (0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, 0, \ldots, 0)$$

with $\mathbf{W}_{k+1}^1$ a translation by two time-units of $\mathbf{W}_k^1$ and each fluctuation value computed as $d_k = \mathbf{f} \cdot \mathbf{W}_k^1$.

A major strength of the Daub 9/7 system is that the analysis wavelet numbers satisfy the following equations (to an accuracy of 10^{-9}):

$$\begin{aligned}
\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 + \beta_6 + \beta_7 &= 0 \\
(-3)\,\beta_1 + (-2)\,\beta_2 + (-1)\,\beta_3 + 0\,\beta_4 + 1\,\beta_5 + 2\,\beta_6 + 3\,\beta_7 &= 0 \\
(-3)^2\,\beta_1 + (-2)^2\,\beta_2 + (-1)^2\,\beta_3 + 0^2\,\beta_4 + 1^2\,\beta_5 + 2^2\,\beta_6 + 3^2\,\beta_7 &= 0 \\
(-3)^3\,\beta_1 + (-2)^3\,\beta_2 + (-1)^3\,\beta_3 + 0^3\,\beta_4 + 1^3\,\beta_5 + 2^3\,\beta_6 + 3^3\,\beta_7 &= 0.
\end{aligned}$$

© 2008 by Taylor & Francis Group, LLC

These equations imply that whenever a signal's values are closely approximated by either a constant sequence, a linear sequence, a quadratic sequence, or a cubic sequence, over the support of a Daub 9/7 analyzing wavelet, then the fluctuation value produced by the scalar product of that wavelet with the signal will closely approximate 0. For many signals that produces a huge number of very small fluctuation values, a crucial property for compression and denoising. The even symmetric extension of signals at their endpoints is also much more likely to allow for close approximation by degree 3 or smaller polynomials than the periodic extension typically employed with the orthogonal wavelets. This feature is very important in image processing where there are lots of endpoints at the boundaries of the images.

Furthermore, the analysis scaling numbers satisfy (to 10^{-9} accuracy):

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 + \alpha_9 = \sqrt{2}$$

and

$$\begin{aligned}(-4)\,\alpha_1 + (-3)\,\alpha_2 + (-2)\,\alpha_3 + (-1)\,\alpha_4 \\ + 0\,\alpha_5 + 1\,\alpha_6 + 2\,\alpha_7 + 3\,\alpha_8 + 4\,\alpha_9 = 0\end{aligned}$$

which imply that the trend values at any given level are often close matches of an analog signal ($\mathbf{a}^1_m \approx \sqrt{2}g(t_{2m})$, $\mathbf{a}^2_m \approx 2g(t_{4m})$, and so on), the same as with coiflets. That property makes for easy interpretation of trend values, especially in image processing.

As with the Daub 5/3 case, the synthesis scaling signals and synthesis wavelets are computed using the inverse Daub 9/7 transform on standard basis vectors. Two examples of these synthesis signals are shown in Figure 3.16. Notice that they are symmetric and much smoother (no sharp corners) than the corresponding Daub 5/3 signals in Figure 3.15(c) and (d). This symmetry

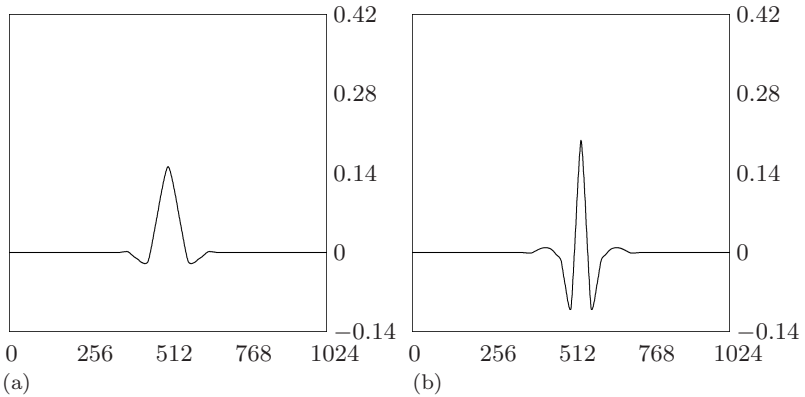


FIGURE 3.16
(a) Daub 9/7 synthesis scaling signal $\widetilde{\mathbf{V}}^6_9$. (b) Daub 9/7 synthesis wavelet $\widetilde{\mathbf{W}}^6_9$. Compare with Figure 3.15.

© 2008 by Taylor & Francis Group, LLC

and smoothness are two of the features that make Daub 9/7 wavelets very useful in image compression and denoising.

The Daub 9/7 system is biorthogonal rather than orthogonal. However, the Daub 9/7 transform is very close to energy preserving. On average, the ratio of $\mathcal{E}_{(\mathbf{a}^1 \mid \mathbf{d}^1)}$ to $\mathcal{E}_{\mathbf{f}}$ is about 1.02, only a 2% difference. This implies that, in applications like compression and denoising, the Daub 9/7 system can be applied *as if* it were orthogonal (energy preserving). In particular, the thresholding methods described previously can still be employed with Daub 9/7 transforms.

3.9 Notes and references

Descriptions of the many other types of wavelets, as well as further mathematical theory and applications, can be found in [2] to [9]. The book [10] is particularly good on the topic of spline wavelets. A lot of fundamental research on spline wavelets and their applications can be found at Michael Unser's webpage [11]. For example, [12] explores the relation between spline wavelets and wavelets in general, and [13] uses the results of [12] to understand the properties of the Daub 5/3 and Daub 9/7 wavelets.

Further discussions of quantization can be obtained from [4], [14], or [15]. Some good references on information theory are [16] to [18]. Further references for arithmetic coding can be found in Section 4.10.

Introductory treatments of noise can be found in [19] and [20]. The method of choosing denoising thresholds for wavelet transforms has received a very complete analysis in the papers [21] and [22], and is given a good summary in [4]. There is also an excellent survey of various wavelet-based noise removal techniques in [23].

Biorthogonal wavelets are given a thorough and accessible introduction in a paper of Sweldens [24]. An excellent treatment of Daub 9/7 transforms and their inversion can be found in the papers of Sweldens and his collaborators, [25] and [26]. The Daub 9/7 system was introduced in [27]. The Daub 5/3 wavelets are also known as Legall 5/3 wavelets [28].

1. Y. Meyer. (1993). *Wavelets. Algorithms and Applications.* SIAM, Philadelphia, PA.

2. C.S. Burrus, R.H. Gopinath, and H. Guo. (1998). *Introduction to Wavelets and Wavelet Transforms, A Primer.* Prentice Hall, Englewood Cliffs, NJ.

3. I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

4. S. Mallat. (1999). *A Wavelet Tour of Signal Processing. Second Edition.* Academic Press, New York, NY.

5. H.L. Resnikoff and R.O. Wells, Jr. (1998). *Wavelet Analysis. The Scalable Structure of Information.* Springer, New York, NY.

6. G. Strang and T.Q. Nguyen. (1996). *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Boston, MA.

© 2008 by Taylor & Francis Group, LLC

7. M. Frazier. (2001). *An Introduction to Wavelets Through Linear Algebra.* Springer, New York, NY.
8. O. Bratteli and P. Jorgensen. (2002). *Wavelets through a Looking Glass.* Birkhäuser, Boston, MA.
9. D.F. Walnut. (2002). *An Introduction to Wavelet Analysis.* Birkhäuser, Boston.
10. C.K. Chui. (1997). *Wavelets: A Mathematical Tool for Signal Analysis.* SIAM, Philadelphia, PA.
11. Michael Unser's webpage: http://bigwww.epfl.ch/unser/
12. M. Unser and T. Blu. (2003). Wavelet theory demystified. *IEEE Transactions on Signal Processing,* Vol. 51, 470–483.
13. M. Unser and T. Blu. (2003). Mathematical properties of the JPEG2000 wavelet filters. *IEEE Transactions on Image Processing,* Vol. 12, 1080–1090.
14. M. Vetterli and J. Kovačević. (1995). *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ.
15. G. M. Davis and A. Nosratinia. (1998). Wavelet-based image coding: an overview. *Applied and Computational Control, Signals and Circuits,* Vol. 1, 205–269.
16. R.B. Ash. (1990). *Information Theory.* Dover, New York, NY.
17. T.M. Cover and J.A. Thomas. (1991). *Elements of Information Theory.* Wiley, New York, NY.
18. D. Hankerson, G.A. Harris and P.D. Johnson, Jr. (2003). *Introduction to Information Theory and Data Compression, Second Edition.* CRC Press, Boca Raton, FL.
19. S.I. Baskakov. (1986). *Signals and Circuits.* Mir, Moscow.
20. R.L. Fante. (1988). *Signal Analysis and Estimation.* Wiley, New York, NY.
21. D. Donoho, et al. (1995). Wavelet shrinkage: asymptopia? *J. of Royal Stat. Soc. B.,* Vol. 57, 301–369.
22. D. Donoho and I. Johnstone. (1994). Ideal spatial adaptation via wavelet shrinkage. *Biometrika,* Vol. 81, 425–455.
23. D. Donoho. (1993). Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. In I. Daubechies, Editor. *Different Perspectives on Wavelets.* AMS, Providence, RI, 173–205.
24. W. Sweldens. (1996). The lifting scheme: a custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis,* Vol. 3, 186–200.
25. A. R. Calderbank, et al. (1998). Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis,* Vol. 5, 332–369.
26. I. Daubechies and W. Sweldens. (1998). Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.,* Vol. 4, 247–269.
27. A. Cohen, I. Daubechies, and J.-C. Feauveau. (1992). Biorthogonal bases of compactly supported wavelets. *Commun. on Pure and Appl. Math.,* Vol. 45, 485–560.

© 2008 by Taylor & Francis Group, LLC

28. D. Legall and A. Tabatai. (1988). Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Int. Conf. Acoustic, Speech, Signal Processing,* New York, 761–765.

3.10 Examples and exercises

Section 3.1

Example 3.1.A [**Figure 3.1**] To produce Figure 3.1(a) we plotted the formula `20x^2(1-x)^4cos(12pi x)` over the interval $[0, 1]$ using 1024 points. Figure 3.1 was then created by choosing *Transform/Wavelet*, and selecting the *Daub 4* option with 2 levels. Figures 3.1(c) and (d) were plotted by selecting *View/Display style* from the menu for the graph of the original signal and changing the X and Y intervals to the ones shown.

Example 3.1.B [**Figure 3.2**] The graphs of $\mathbf{V}_1^5$, $\mathbf{V}_8^5$, and $\mathbf{V}_{16}^5$ were produced in the following way. To produce $\mathbf{V}_1^5$ we applied the 5^{th} level inverse Daub4 transform to the signal $(1, 0, 0, \ldots, 0)$. This signal was produced by plotting the formula `del(x)` over the interval $[0, 1024]$ with 1024 points. We then chose *Transform/Wavelet* and selected the *Daub 4* option, with 5 levels and the *Inverse* box checked. To produce $\mathbf{W}_1^5$ we applied the 5^{th} level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the 33^{rd} position (that signal was produced by plotting the formula `del(x-32)` over the interval $[0, 1024]$ with 1024 points). The reason this works is explained in detail in subsection 3.7.4. As explained in that subsection, we produce $\mathbf{V}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the k^{th} position, $k = 1, \ldots, N/2^m$; and we produce $\mathbf{W}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the $k + N/2^m$ position, $k = 1, \ldots, N/2^m$.

Example 3.1.C [**Figure 3.3**] Figure 3.3 was produced in the same way as Figure 2.3 (see Example 2.4.D), except that Daub4 was used as wavelet choice.

3.1.1$_{\mathbf{s}}$ Explain why $\mathbf{V}_m^2$ has a support of 10 time-units and is a translate of $\mathbf{V}_1^2$ by $4(m-1)$ time-units. (Ignore wrap-around.)

3.1.2 The 3^{rd} level Daub4 scaling signals and wavelets have supports of how many time-units? Are they all shifts of $\mathbf{V}_1^3$ and $\mathbf{W}_1^3$?

3.1.3$_{\mathbf{s}}$ Does $0\alpha_1 + 1\alpha_2 + 2\alpha_3 + 3\alpha_4 = 0$ hold for the Daub4 scaling numbers?

3.1.4 Show that Property I holds for 2-level wavelets, i.e., *if a signal* $\mathbf{f}$ *is (approximately) linear over the support of a* 2*-level Daub4 wavelet* $\mathbf{W}_m^2$*, then the* 2*-level fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^2$ *is (approximately) zero.*

3.1.5$^{\mathbf{c}}$ Plot 1-level Daub4 transforms of the following functions—sampled uniformly over $[0, 1]$ using 1024 points. [*Note:* These are the same functions considered in problem 2.1.5.]

(a) $f(x) = x^2(1-x)$

(b)$_{\mathbf{s}}$ $f(x) = x^4(1-x)^6 \cos 64\pi x$

© 2008 by Taylor & Francis Group, LLC

(c) $(0.2 < x < 0.3) - 3(0.4 < x < 0.5) + 2(0.5 < x < 0.8)$

(d) $f(x) = \operatorname{sgn}(\sin 12\pi x)$

Remark In order to use FAWAV to graph the averaged signals—$\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, etc.—you proceed as follows. After plotting the signal $\mathbf{f}$, you choose *Series/Wavelet* and select *Ascending terms* as the series choice. If you specify $N/2$ number of terms (where N is the number of points), then the first averaged signal $\mathbf{A}^1$ will be plotted. Or, if you select $N/4$ number of points, then $\mathbf{A}^2$ will be plotted. Or, by selecting $N/8$ number of points, then $\mathbf{A}^3$ will be plotted. (This method of plotting averaged signals is equivalent to taking a wavelet transform, setting all values of the transform to 0 for indices above $N/2$, $N/4$, or $N/8$, and then inverse transforming.)

3.1.6$^{\mathbf{c}}$ Compute the Daub4 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.1.7$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.1.6 and the given signal?

3.1.8$^{\mathbf{c}}$ Repeat problems 3.1.6 and 3.1.7 for the signal

$$g(x) = 20x^2(1-x)^2 \cos 64\pi x + 30x^2(1-x)^4 \sin 30\pi x.$$

3.1.9 Show that $\mathbf{f} \cdot \mathbf{W}_m^1 = \mathrm{O}(h)$ when $\mathbf{W}_m^1$ is a 1-level Haar wavelet. [*Hint:* Use Formula (3.15).]

3.1.10$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 2.5.2, but use a Daub4 transform instead of a Haar transform.

3.1.11$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub4 series instead of a Haar series. Which function is best approximated by a Daub4 series? Why?

Section 3.2

Example 3.2.A [Figure 3.4] Figure 3.4 was created by first plotting the function

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

over the interval $[0, 2]$ using 4096 points. Then, the graph in (a) was created by selecting *Transform/Wavelet* and choosing a Haar wavelet with 2 levels. The corresponding cumulative energy profile in (c) was created by right-clicking on the Haar transform graph and selecting *Energy graph.* The graphs in (b) and (d) were created in a similar way, except that a Daub4 transform was used.

3.2.1 Verify Equations (3.17a)–(3.17c).

3.2.2$^{\mathbf{c}}_{\mathbf{s}}$ Compute the 3-level Daub4 transform of the signal $\mathbf{f}$ obtained from 1024 uniformly spaced samples of

$$g(x) = x^2(4-x)^4 \sin 12\pi x$$

© 2008 by Taylor & Francis Group, LLC

over the interval $[0, 4]$. Compute the energy of $\mathbf{f}$ and of its transform and check that conservation of energy holds.

3.2.3$^{\mathbf{c}}$ Using $\epsilon = 0.0001\,\mathcal{E}_{\mathbf{f}}$, compute the percentage of 1-level Daub4 transform values which are less than ϵ for each of the signals in problem 2.1.5 (a)–(d). Compare your results with a 1-level Haar transform [see problem 2.2.6].

3.2.4$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 3.2.3, but use a 2-level Daub4 transform instead. Compare your results with a 2-level Haar transform.

Section 3.3

Example 3.3.A [Figure 3.5] The signal analyzed in Figure 3.5 was created by plotting the formula

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

over the interval $[0, 2]$ using 4096 points. To compute 1000 times its fluctuation $\mathbf{d}^1$, we plotted the 1-level Daub4 transform and then graphed

```
1000g1(x/2+1)
```

To compute 30 times the fluctuation of its fluctuation $\mathbf{d}^3$ we plotted its 3-level Daub4 transform and then graphed

```
30g1(x/8+1/4)
```

Similar computations were done for the Daub6 case.

Example 3.3.B [Figure 3.6] Figure 3.6 was produced in the same way as Figure 2.3 (see Example 2.4.D), except Daub20 was used as the choice of wavelet.

Example 3.3.C [Figure 3.7] Figure 3.7 was produced in the same way as Figure 3.2 (see Example 3.1.B), except an inverse Coif6 transform was used.

3.3.1$_{\mathbf{s}}$ Show that if $\mathbf{f}$ is obtained from samples of a 3-times continuously differentiable function g over the support of a 1-level Daub6 wavelet $\mathbf{W}^1_m$, then the fluctuation value $\mathbf{f} \cdot \mathbf{W}^1_m$ satisfies $\mathbf{f} \cdot \mathbf{W}^1_m = \mathrm{O}(h^3)$.

3.3.2$^{\mathbf{c}}$ Compute the Daub6 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.3.3$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.3.2 and the original signal?

3.3.4$^{\mathbf{c}}$ Repeat problems 3.3.2 and 3.3.3 for the signal

$$g(x) = 20x^2(1-x)^2 \cos 64\pi x + 30x^2(1-x)^4 \sin 30\pi x.$$

3.3.5$^{\mathbf{c}}$ Repeat problem 2.5.2, but use a Daub6 wavelet series instead of a Haar series.

© 2008 by Taylor & Francis Group, LLC

3.3.6$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub6 wavelet series instead of a Haar series. Which function is best approximated by a Daub6 series? Why?

3.3.7$^{\mathbf{c}}$ Repeat problem 2.5.2, but use a Daub8 wavelet series instead of a Haar series.

3.3.8$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub8 wavelet series instead of a Haar series. Which function is best approximated by a Daub8 series? Why?

3.3.9$^{\mathbf{c}}_{\mathbf{s}}$ For the following function, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub$J$ series for each $J = 4, 6, 8$ and for each level $L = 1, 2, \ldots, 6$ (using 1024 points over the interval $[0, 1]$):

$$g(x) = x^2(1-x)^6 \cos 25\pi x.$$

Note: To determine the minimum number of terms you use the option *Energy fraction* for a wavelet series and enter the value 0.9999 to require 99.99% of the energy. A report is displayed that indicates the number of terms used.

3.3.10$^{\mathbf{c}}$ Repeat problem 3.3.9 for the Daub$J$ series, $J = 10, 12, 14$.

3.3.11$^{\mathbf{c}}$ Repeat problem 3.3.9 for each of the following functions (and use Haar series as well as Daub$J$ series):

(a) $g(x) = [0.1 < x < 0.2] - 3[0.3 < x < 0.5] + 9[0.7 < x < 0.9]$

(b) $g(x) = x[0.1 < x < 0.2] - x^2[0.3 < x < 0.5] + 9[0.7 < x < 0.9]$

(c) $g(x) = [0.1 < x < 0.2] \cos 40\pi x - [0.3 < x < 0.5] \sin 30\pi x + [0.7 < x < 0.9] \cos 40\pi x$

(d) $g(x) = [0.1 < x < 0.2] \sin 40\pi x + [0.3 < x < 0.5] 2 \sin 30\pi x + [0.7 < x < 0.9] \sin 60\pi x$

3.3.12$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 3.3.9, but use Coif$I$ series for $I = 6, 18, 30$.

3.3.13$^{\mathbf{c}}$ Repeat problem 3.3.11, but use Coif$I$ series for $I = 6, 18, 30$.

3.3.14$^{\mathbf{c}}$ Compare 3-level Coif6 trend values with $2\sqrt{2}g(8x)$ over the interval $[0, 0.125]$, where $g(x)$ is defined by Equation (3.35) [use 2^{14} samples over the interval $[0, 1]$ as in the text]. Do the same for the Daub4 transform. *Note:* By "compare" we mean determine the maximum error.

3.3.15$^{\mathbf{c}}$ For the following function $g(x)$ over the interval $[0, 1]$:

$$g(x) = 40x^4(1-x)^6 \cos 24\pi x$$

compare $\sqrt{2}g(2x)$ over the interval $[0, 0.5]$ with the 1-level Coif$I$ trend values for $I = 6, 12, 18, 24, 30$. [Use 2^{12} samples, and by "compare" we mean find the maximum error.] Do the same for Daub$J$ trend values for $J = 6, 12, \ldots, 18$.

3.3.16$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 3.3.15, but compare $2g(4x)$ over $[0, 0.25]$ with Coif$I$ and Daub$J$ 2-level trend values.

© 2008 by Taylor & Francis Group, LLC

Section 3.4

Example 3.4.A [Figure 3.8] Figure 3.8 is produced in the same way as Figure 2.5 (see Example 2.5.C), except that a Coif30 transform is used.

3.4.1$^{\mathbf{c}}$ Produce graphs like the ones shown in Figures 2.4 and 3.8 for 1024 samples of the following signals over the interval $[0, 10]$:

(a) $f(x) = x$

(b) $f(x) = 2[2 < x < 4] - 2[5 < x < 7] + 2[8 < x < 9]$

(c) $f(x) = 2[2 < x < 4] - x[5 < x < 7] + 2[8 < x < 9]$

(d)$_{\mathbf{s}}$ $f(x) = 0.001x^4(10 - x)^2$

What thresholds should be used to capture 99.99% of the energy using a 10-level Daub4 wavelet transform? What compression ratios do these produce, and what are the maximum errors between the original signals and the compressed signals?

3.4.2$^{\mathbf{c}}$ Repeat problem 3.4.1, but use a 10-level Daub18 transform.

3.4.3$^{\mathbf{c}}$ Repeat problem 3.4.1, but use a 10-level Coif12 transform.

3.4.4$^{\mathbf{c}}$ Repeat problem 3.4.1, but use a 10-level Coif30 transform.

3.4.5$^{\mathbf{c}}_{\mathbf{s}}$ For each of the functions in problem 3.4.1, find the level (from 1 to 10) which uses the least number of Daub4 transform values to capture 99.99% of each signal's energy. [*Hint:* Use the *energy percentage* method for wavelet series.]

3.4.6$^{\mathbf{c}}$ Repeat problem 3.4.5, but use a Daub12 transform.

3.4.7$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 3.4.5, but use a Coif18 transform.

3.4.8$^{\mathbf{c}}$ Repeat problem 3.4.5, but use a Coif30 transform.

3.4.9$^{\mathbf{c}}$ For the audio signal `alfalfa_2.wav`, determine the level of a Coif30 series that uses the least number of transform values to capture 99.99% of the energy.

3.4.10$^{\mathbf{c}}$ Repeat problem 3.4.9, but with the recording `brazil_clip.wav`.

3.4.11$^{\mathbf{c}}$ Repeat problem 3.4.9, but with the recording `house_wren_chirp.wav`.

3.4.12$^{\mathbf{c}}$ Repeat problem 3.4.9, but with the recording `Sara's 'Bait'.wav`.

Section 3.5

Example 3.5.A [Figure 3.10] To graph the signal in (a) you select *New 1-dim* from the menu, right-click on the graph area and select *Load/Sound file,* and select `greasy.wav` as the sound file to load. To plot the histogram in (b) you right-click on the graph and select *Histogram.* You then choose an 8-bit histogram with both of the choices *Include zero values* and *Include sign bit* checked. To obtain the graph in (c) you perform a 14-level Coif30 transform of the signal in (a). Finally, to get the graph in (d) you compute a histogram of the transform, but uncheck the choice *Include zero values* and make sure that the choice *Include sign bit* is checked. (We call this a *dead-zone histogram.*)

© 2008 by Taylor & Francis Group, LLC

Example 3.5.B [Figure 3.11] To obtain the plot of the fourth trend in (a), you plot a 4-level Coif30 transform of the `greasy.wav` signal, select *View/Display,* and plot over the new X-interval: `0,.743038548752834/2^4`. To obtain the graph in (b) you plot a histogram with the *Include zero values* unchecked, and the choice *Include sign bit* checked. To obtain the graph in (c), you change the X-interval of the Coif30 transform to

```
0.743038548752834/2^4,0.743038548752834.
```

To obtain the histogram in (d) you compute a dead-zone histogram of the graph from (c).

3.5.1^{c} Draw a plot of the *entropy function,* $f(x) = x \log_2(1/x)$, over the interval $[0, 1]$. [*Hint:* In FAWAV, there is a built-in function, entr(x), which calculates $x \log_2(1/x)$.] Find the point where the maximum of this function lies, either by numerical estimation (using the *Analysis/Trace* procedure) or by calculus.

3.5.2$_{s}$ Show that the entropy of the uniform probability sequence, $p_k = 1/N$ for $k = 1, 2, \ldots, N$, is $\log_2 N$.

3.5.3 Find the entropy of the sequence $p_k = c\,2^{-k}$, $k = 1, 2, \ldots, 16$ (where $c = 1/\sum_{k=1}^{16} 2^{-k}$).

3.5.4$^{c}_{s}$ Using the recording `alfalfa_2.wav`, estimate—as was done in the text for *greasy*—the number of bpp needed for an optimal, entropy-based compression of this 16 bpp recording, then compare this with a maximum-level Coif30 transform at 16 bpp, and with a 4-level Coif30 transform at 16 bpp for the trend and 12 bpp for the fluctuations.

3.5.5^{c} Repeat problem 3.5.4, but use the recording `alfalfa_22050.wav`.

3.5.6$^{c}_{s}$ For problems 3.5.4 and 3.5.5, calculate the Sup-norm and relative 2-norm errors between the original and compressed signals. [*Note:* The compressed signals can be plotted using *Thresholded wavelet series,* when the settings are adjusted (press the button labeled *Edit settings*) to *Quantized thresholding* and either *Uniform threshold* or *Multiple thresholds* are selected.]

Section 3.6

Example 3.6.A [Figure 3.12] For the graph in Figure 3.12(a), we plotted

```
0.1ran(x)+40x^2(1-x)^4cos(12pi x)(0<x<1)+{40(x-1)^2(2-x)^8cos(48pi x)
    +80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

over the interval $[0, 2]$ using 4096 points. The graph in (b) was obtained by a 12-level Coif30 transform of this signal, and then changing to the Y-interval $[-2, 2]$. By right-clicking on the graph and selecting *Display cursor coordinates* we were able to scan over the graph and obtain 0.2 as a threshold for eliminating noisy transform values. To obtain the plot in (c) we plotted the function

```
g1(x)(abs(g1(x))>0.2)
```

© 2008 by Taylor & Francis Group, LLC

where `g1(x)` stands for the transform values. Then by applying an inverse Coif30 transform to the graph in (c) we obtained the graph shown in (d), the denoised signal.

Example 3.6.B [**Figure 3.13**] To produce the graph in Figure 3.13(a), we plotted the function `0.5rang(0)` over $[0, 1]$ using 4096 points. The mean and standard deviation of this noise were found by selecting *Analysis/Statistics* from the menu. To graph the histogram in (b) we calculated an 8-bit histogram with both options (*Include zero values* and *Include sign bit*) checked. To obtain the graph in (c) we performed a 12-level Coif30 transform of the noise signal from (a). We then calculated a histogram [the same type as we did with (a)] to get the graph in (d).

Example 3.6.C [**Figure 3.14**] To get the graph in (a) we loaded the sound file `noisy_wolf_whistle.wav`. The graph in (b) was obtained by performing a 15-level Coif18 transform of this sound file. To obtain the graph in (c) we then processed this transform by selecting *Graph/Plot* and plotting the formula contained in the file `fig_3_14_(c).uf1` (by clicking on the *Load* button under the formula text area). This is one of the files that is contained in the zip file `Primer_Figures.zip` at the book's website. After producing the processed transform in (c) we then performed an inverse transform on it to produce the denoised signal in (d).

3.6.1^{c} Using the threshold method, denoise each of the following signals (each of which is defined over the interval $[0, 10]$ using 1024 points). Use a 10-level Coif30 transform.

(a) `20(2 < x < 4) - 30(5<x<7) + 40(8<x<9) + 5rang(0)`

(b)$_{s}$ `40cos(4pi x) + 5rang(0)`

(c) `20[2< x < 4] + 5x[5 < x < 7] + 20[8 < x < 9]+ 5rang(0)`

(d) `40cos(4pi x)(2 < x < 6) + 5rang(0)`

Which denoisings would you deem to be the most successful, and why?

3.6.2^{c} Repeat problem 3.6.1, but use a 4-level Coif30 transform.

3.6.3^{c} Repeat problem 3.6.1, but use a 10-level Coif18 transform and also a 4-level Coif18 transform.

Remark In Exercises 3.6.4 to 3.6.6, you should use the built-in wavelet denoising method obtained by selecting *Denoise/Wavelet* and checking the box labelled *Average.* Try 5-levels for the wavelet transform. This procedure automatically selects the threshold and performs an average of denoisings of shiftings of the signal (this further reduces noise).

3.6.4^{c} Accompanying these exercises is a data file, `noisy word 1.wav`, which is a noisy version of the audio file `alfalfa_2.wav`. Use wavelet-based denoising to denoise this signal. What percentage reduction of RMS do you obtain?

3.6.5^{c} Repeat problem 3.6.4, but for the audio file `noisy word 2.wav.`

3.6.6^{c} Repeat problem 3.6.4, but for the audio file `noisy word 3.wav.`

© 2008 by Taylor & Francis Group, LLC

Section 3.7

Example 3.7.A Find the Daub 5/3 transform of $\mathbf{f} = (8, 16, 8, -8, 0, 16)$.

Solution. Using the formulas for the analysis scaling vectors $\{\mathbf{V}_k\}$ and the equation $a_k = \mathbf{f} \cdot \mathbf{V}_k$ we obtain

$$\begin{aligned} a_1 &= (8,16,8,-8,0,16) \cdot (\tfrac{3}{4}, \tfrac{1}{2}, \tfrac{-1}{4}, 0, 0, 0) \;= 12 \\ a_2 &= (8,16,8,-8,0,16) \cdot (\tfrac{-1}{8}, \tfrac{1}{4}, \tfrac{3}{4}, \tfrac{1}{4}, \tfrac{-1}{8}, 0) \;= 7 \\ a_3 &= (8,16,8,-8,0,16) \cdot (0, 0, \tfrac{-1}{8}, \tfrac{1}{4}, \tfrac{5}{8}, \tfrac{1}{4}) \;= 1. \end{aligned}$$

Similarly, we compute d_k from $d_k = \mathbf{f} \cdot \mathbf{W}_k$ where $\{\mathbf{W}_k\}$ are the analysis wavelets:

$$\begin{aligned} d_1 &= (8,16,8,-8,0,16) \cdot (\tfrac{-1}{2}, 1, \tfrac{-1}{2}, 0, 0, 0) \;= 8 \\ d_2 &= (8,16,8,-8,0,16) \cdot (0, 0, \tfrac{-1}{2}, 1, \tfrac{-1}{2}, 0) \;= -12 \\ d_3 &= (8,16,8,-8,0,16) \cdot (0,0,0,0,-1,1) \;= 16. \end{aligned}$$

So the 1-level Daub 5/3 transform is $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (12, 7, 1 \,|\, 8, -12, 16)$.

Example 3.7.B [Figure 3.15] As explained in subsection 3.7.4, we produce $\widetilde{\mathbf{V}}_k^m$ by applying the m-level inverse Daub 5/3 transform (select `DD 5/3 (2,2)` as transform type) to the signal $(0, \ldots, 0, \underline{1}, 0, \ldots, 0)$ where the 1 is in the k^{th} position, $k = 1, \ldots, N/2^m$; and we produce $\widetilde{\mathbf{W}}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the $k + N/2^m$ position, $k = 1, \ldots, N/2^m$.

3.7.1$^{\mathbf{c}}$ Compute the Daub 5/3 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos(48\pi x)$$

over the interval $[0, 1]$ using 1024 points.

3.7.2$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.7.1 and the original signal?

3.7.3$^{\mathbf{c}}$ For the signal

$$g(x) = 20x^2(1-x)^2 \cos(64\pi x) + 30x^2(1-x)^4 \sin(30\pi x)$$

repeat problems 3.7.1 and 3.7.2.

3.7.4$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub 5/3 wavelet series instead of a Haar series. Which function is best approximated by a Daub 5/3 series? Why?

3.7.5$^{\mathbf{c}}_{\mathbf{s}}$ For the function

$$g(x) = x^2(1-x)^6 \cos(25\pi x)$$

graphed over the interval $[0, 1]$ using 1024 points, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub 5/3 series for each level $L = 1, 2, \ldots, 6$.

© 2008 by Taylor & Francis Group, LLC

3.7.6$^{c}_{s}$ Compare 1-level Daub 5/3 trend values with $g(2x)$ over the interval $[0, 0.5]$, where $g(x)$ is defined by Equation (3.35). Use 2^{14} samples over the interval $[0, 1]$. *Note:* By "compare" we mean determine the maximum error.

3.7.7^{c} Compare 2-level Daub 5/3 trend values with $g(4x)$, over $[0, 0.25]$, where $g(x)$ is defined by Equation (3.35). Also compare 3-level Daub 5/3 trend values with $g(8x)$ over the interval $[0, 0.125]$. Use 2^{14} samples over the interval $[0, 1]$.

Section 3.8

Example 3.8.A [Figure 3.16] As explained in subsection 3.7.4, we produce $\widetilde{\mathbf{V}}_k^m$ by applying the m-level inverse Daub 9/7 transform (select `Daub 9/7` as the transform type) to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the k^{th} position, $k = 1, \ldots, N/2^m$; and we produce $\widetilde{\mathbf{W}}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the $k + N/2^m$ position, $k = 1, \ldots, N/2^m$.

3.8.1^{c} Compute the Daub 9/7 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos(48\pi x)$$

over the interval $[0, 1]$ using 1024 points.

3.8.2$^{c}_{s}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.8.1 and the original signal?

3.8.3^{c} For the signal

$$g(x) = 20x^2(1-x)^2 \cos(64\pi x) + 30x^2(1-x)^4 \sin(30\pi x)$$

repeat problems 3.8.1 and 3.8.2.

3.8.4^{c} Repeat problem 2.5.3, but use a Daub 9/7 wavelet series instead of a Haar series. Which function is best approximated by a Daub 9/7 series? Why?

3.8.5$^{c}_{s}$ For the function

$$g(x) = x^2(1-x)^6 \cos(25\pi x)$$

graphed over the interval $[0, 1]$ using 1024 points, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub 9/7 series for each level $L = 1, 2, \ldots, 6$.

3.8.6$^{c}_{s}$ Compare 1-level Daub 9/7 trend values with $\sqrt{2}g(2x)$ over the interval $[0, 0.5]$, where $g(x)$ is defined by Equation (3.35). Use 2^{14} samples over the interval $[0, 1]$. *Note:* By "compare" we mean determine the maximum error.

3.8.7^{c} Compare 2-level Daub 9/7 trend values with $2g(4x)$ over the interval $[0, 0.25]$, where $g(x)$ is defined by Equation (3.35). Also compare 3-level Daub 5/3 trend values with $g(8x)$ over the interval $[0, 0.125]$. Use 2^{14} samples over the interval $[0, 1]$.

© 2008 by Taylor & Francis Group, LLC

Chapter 4

Two-dimensional wavelets

Optimal processing of an image... requires the representation of an image at different scales. In order to meet this demand, we need a multiscale representation *of images.*

—Bernd Jähne[1]

Until now we have been working with one-dimensional signals, but wavelet analysis can be done in any number of dimensions. The essential ideas, however, are revealed in two dimensions. In recent years wavelet methods have come to predominate in advanced techniques in image processing. We shall provide a basic summary of the success stories from this field. Some of the highlights of the chapter include descriptions of two state of the art wavelet-based image compression algorithms, including the new JPEG 2000 standard, and description of a high performance image denoising algorithm. We shall also describe some aspects of image enhancement and analysis.

4.1 Two-dimensional wavelet transforms

In this section we shall begin our treatment of 2D wavelet analysis. Many of the basic ideas are similar to the 1D case; so we shall not repeat similar formulas but rather focus on the new ideas that are needed for the 2D case. We begin with a definition of discrete images, the types of images that we shall be working with.

[1] Jähne's quote is from [1].

© 2008 by Taylor & Francis Group, LLC

4.1.1 Discrete images

A *discrete image* $\mathbf{f}$ is an array of M rows and N columns of real numbers:

$$\mathbf{f} = \begin{pmatrix} f_{1,M} & f_{2,M} & \cdots & f_{N,M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1,2} & f_{2,2} & \cdots & f_{N,2} \\ f_{1,1} & f_{2,1} & \cdots & f_{N,1} \end{pmatrix}. \tag{4.1}$$

The *values* of $\mathbf{f}$ are the MN real numbers $\{f_{j,k}\}$. It should be noted that the way in which the values of $\mathbf{f}$ are displayed in the array on the right side of (4.1) is not the most commonly used one. We chose to display the values of $\mathbf{f}$ in this way because it corresponds well with the case where $\mathbf{f}$ is an array of sample values:

$$\mathbf{f} = \begin{pmatrix} g(x_1, y_M) & g(x_2, y_M) & \cdots & g(x_N, y_M) \\ \vdots & \vdots & \ddots & \vdots \\ g(x_1, y_2) & g(x_2, y_2) & \cdots & g(x_N, y_2) \\ g(x_1, y_1) & g(x_2, y_1) & \cdots & g(x_N, y_1) \end{pmatrix} \tag{4.2}$$

of a function $g(x, y)$ at the sample points (x_j, y_k) in the Cartesian coordinate plane. Just as with discrete 1D signals, it is frequently the case that a discrete 2D image is obtained from samples of some function $g(x, y)$.

It is often helpful to view a discrete image in one of two other ways. First, as a single column consisting of M signals having length N,

$$\mathbf{f} = \begin{pmatrix} \mathbf{f}_M \\ \vdots \\ \mathbf{f}_2 \\ \mathbf{f}_1 \end{pmatrix} \tag{4.3}$$

with the rows being the signals

$$\begin{aligned} \mathbf{f}_M &= (f_{1,M}, f_{2,M}, \ldots, f_{N,M}) \\ &\vdots \\ \mathbf{f}_2 &= (f_{1,2}, f_{2,2}, \ldots, f_{N,2}) \\ \mathbf{f}_1 &= (f_{1,1}, f_{2,1}, \ldots, f_{N,1}). \end{aligned}$$

Second, as a single row consisting of N signals of length M, written as columns,

$$\mathbf{f} = (\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^N) \tag{4.4}$$

with the columns being the signals

$$\mathbf{f}^1 = \begin{pmatrix} f_{1,M} \\ \vdots \\ f_{1,2} \\ f_{1,1} \end{pmatrix}, \mathbf{f}^2 = \begin{pmatrix} f_{2,M} \\ \vdots \\ f_{2,2} \\ f_{2,1} \end{pmatrix}, \ldots, \mathbf{f}^N = \begin{pmatrix} f_{N,M} \\ \vdots \\ f_{N,2} \\ f_{N,1} \end{pmatrix}.$$

© 2008 by Taylor & Francis Group, LLC

Notice that, because of our somewhat peculiar notation, the row index for each column increases from bottom to top (rather than from top to bottom, which is more common notation in image processing).

As an example of the utility of Formulas (4.3) and (4.4), we consider the calculation of the energy of a discrete image. The *energy* $\mathcal{E}_{\mathbf{f}}$ of a discrete image $\mathbf{f}$ is defined to be the sum of the squares of all of its values. Because of (4.3) it follows that $\mathcal{E}_{\mathbf{f}}$ is the sum of the energies of all of the row signals:

$$\mathcal{E}_{\mathbf{f}} = \mathcal{E}_{\mathbf{f}_1} + \mathcal{E}_{\mathbf{f}_2} + \cdots + \mathcal{E}_{\mathbf{f}_M}.$$

Or, because of (4.4), it follows that $\mathcal{E}_{\mathbf{f}}$ is also the sum of the energies of all of the column signals:

$$\mathcal{E}_{\mathbf{f}} = \mathcal{E}_{\mathbf{f}^1} + \mathcal{E}_{\mathbf{f}^2} + \cdots + \mathcal{E}_{\mathbf{f}^N}.$$

From these last two identities it is easy to see when the 2D wavelet transforms defined below have the property of conserving the energy of discrete images.

4.1.2 2D wavelet transforms

A 2D wavelet transform of a discrete image can be performed whenever the image has an even number of rows and an even number of columns.[2] A 1-level wavelet transform of an image $\mathbf{f}$ is defined, using any of the 1D wavelet transforms that we have discussed, by performing the following two steps.

Step 1. Perform a 1-level, 1D wavelet transform, on each row of $\mathbf{f}$, thereby producing a new image.

Step 2. On the new image obtained from Step 1, perform the same 1D wavelet transform on each of its columns.

It is not difficult to show that Steps 1 and 2 could be done in reverse order and the result would be the same. A 1-level wavelet transform of an image $\mathbf{f}$ can be symbolized as follows:

$$\mathbf{f} \longmapsto \begin{pmatrix} \mathbf{h}^1 & | & \mathbf{d}^1 \\ — & & — \\ \mathbf{a}^1 & | & \mathbf{v}^1 \end{pmatrix} \tag{4.5}$$

where the subimages $\mathbf{h}^1$, $\mathbf{d}^1$, $\mathbf{a}^1$, and $\mathbf{v}^1$ each have $M/2$ rows and $N/2$ columns. We shall now discuss the nature of each of these subimages.

The subimage $\mathbf{a}^1$ is created by computing trends along rows of $\mathbf{f}$ followed by computing trends along columns; so it is an averaged, lower resolution version of the image $\mathbf{f}$. For example, in Figure 4.1(a) we show a simple test image of an octagon, and in Figure 4.1(b) we show its 1-level Coif6 transform.

[2] We adopt this convention for simplicity. When an image has an odd number of rows or columns, it can be extended by appending 0's so that it has an even number of rows and columns.

© 2008 by Taylor & Francis Group, LLC

The $\mathbf{a}^1$ subimage appears in the lower left quadrant of the Coif6 transform, and it is clearly a lower resolution version of the original octagon image. Since a 1D trend computation is $\sqrt{2}$ times an average of a small number of successive values in a signal, and the 2D trend subimage $\mathbf{a}^1$ was computed from trends along both rows and columns, it follows that each value of $\mathbf{a}^1$ is equal to 2 times an average of a small square containing adjacent values from the image $\mathbf{f}$. A useful way of expressing the values of $\mathbf{a}^1$ is as scalar products of the image $\mathbf{f}$ with scaling signals, as we did in the 1D case; we shall say more about this later in the section.

The $\mathbf{h}^1$ subimage is created by computing trends along rows of the image $\mathbf{f}$ followed by computing fluctuations along columns. Consequently, wherever there are horizontal edges in an image, the fluctuations along columns are able to detect these edges. This tends to emphasize the horizontal edges, as can be seen clearly in Figure 4.1(b) where the subimage $\mathbf{h}^1$ appears in the upper left quadrant. Furthermore, notice that vertical edges, where the octagon image is constant over long stretches, are removed from the subimage $\mathbf{h}^1$. This discussion should make it clear why we shall refer to this subimage as the first *horizontal fluctuation.*

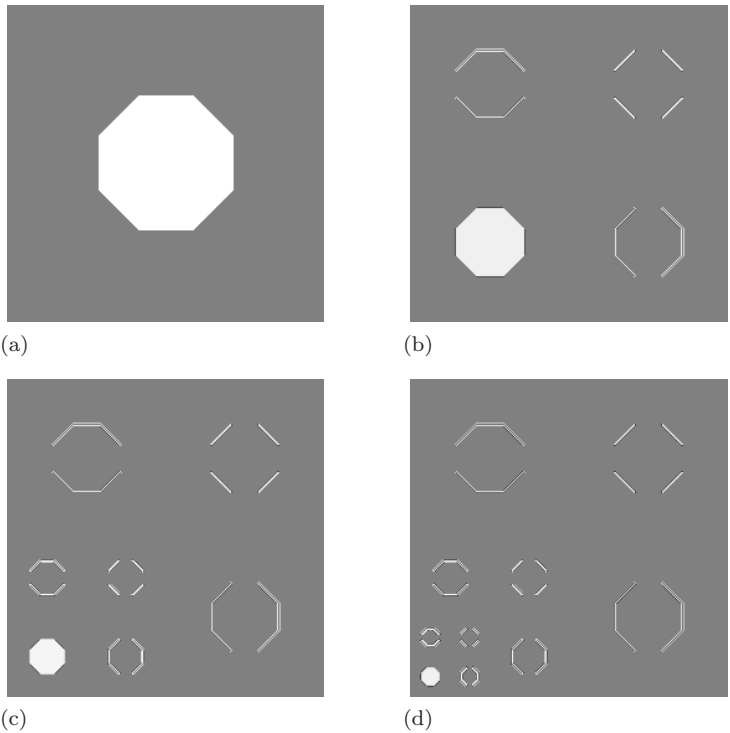


FIGURE 4.1
(a) Octagon image. (b) 1-level Coif6 transform. (c) 2-level Coif6 transform. (d) 3-level Coif6 transform.

© 2008 by Taylor & Francis Group, LLC

The subimage $\mathbf{v}^1$ is similar to $\mathbf{h}^1$, except that the roles of horizontal and vertical are reversed. In Figure 4.1(b) the subimage $\mathbf{v}^1$ is shown in the lower right quadrant. Notice that horizontal edges of the octagon are erased, while vertical edges are emphasized. This is typically the case with $\mathbf{v}^1$, which we shall refer to as the first *vertical fluctuation.*

Finally, there is the first *diagonal fluctuation,* $\mathbf{d}^1$. This subimage tends to emphasize diagonal features, because it is created from fluctuations along both rows and columns. These fluctuations tend to erase horizontal and vertical edges where the image is relatively constant. For example, in Figure 4.1(b) the diagonal fluctuation appears in the upper right quadrant of the image, and it is clear that diagonal details are emphasized while horizontal and vertical edges are erased.

It should be noted that the basic principles discussed previously for 1D wavelet analysis still apply here in the 2D setting. For example, the fact that fluctuation values are generally much smaller than trend values is still true. In the wavelet transform shown in Figure 4.1(b), for instance, the fluctuation subimages $\mathbf{h}^1$, $\mathbf{v}^1$, and $\mathbf{d}^1$ have significantly smaller values than the values in the trend subimage $\mathbf{a}^1$. In fact, in order to make the values for $\mathbf{h}^1$, $\mathbf{v}^1$, and $\mathbf{d}^1$ visible, they are displayed on a logarithmic intensity scale, while the values for the trend subimage $\mathbf{a}^1$ are displayed using an ordinary, linear scale.

Furthermore, 2D wavelet transforms enjoy the conservation of energy property. As noted above, the energy of an image is the sum of the energies of each of its rows or each of its columns. When the 1D wavelet transforms of the rows, performed in Step 1, preserve the row energies, the image obtained in Step 1 will have the same energy as the original image. Likewise, when the 1D wavelet transforms of the columns preserve their energies, it follows that the transform obtained in Step 2 has the same energy as the image from Step 1. Thus the 1-level wavelet transform has the same energy as the original image. For example, the energy of the octagon image in Figure 4.1(a) is 3919.0625, while the energy of its 1-level Coif6 transform is 3919.0622; the slight discrepancy between the two energies is attributable to the inevitable rounding error that arises from finite precision computer calculations.

As in 1D, multiple levels of 2D wavelet transforms are defined by repeating the 1-level transform of the previous trend. For example, a 2-level wavelet transform is performed by computing a 1-level transform of the trend subimage $\mathbf{a}^1$ as follows:

$$\mathbf{a}^1 \longmapsto \begin{pmatrix} \mathbf{h}^2 & | & \mathbf{d}^2 \\ — & & — \\ \mathbf{a}^2 & | & \mathbf{v}^2 \end{pmatrix}.$$

The 1-level fluctuations $\mathbf{h}^1$, $\mathbf{d}^1$, and $\mathbf{v}^1$ remain unchanged. In Figure 4.1(c) we show a 2-level Coif6 transform of the octagon image. In general, a k-level transform is defined by performing a 1-level transform on the previous trend $\mathbf{a}^{k-1}$. In Figure 4.1(d) we show a 3-level Coif6 transform of the octagon image.

It is interesting to compare the successive levels of the Coif6 transforms

© 2008 by Taylor & Francis Group, LLC

in Figure 4.1. Notice how it appears that we are systematically decomposing the original octagon image by peeling off edges; and these edges are retained within the fluctuation subimages. This aspect of wavelet transforms plays a major role in the fields of *image recognition* and *image enhancement.* In Section 4.9 we shall discuss a few examples from these fields.

Besides conservation of energy these 2D wavelet transforms also perform a compaction of energy. For example, for the octagon image in Figure 4.1 most of the energy of the image is successively localized into smaller and smaller trend subimages, as summarized in Table 4.1. Notice, for example, that the third trend $\mathbf{a}^3$, which is 64 times smaller than $\mathbf{f}$ in terms of numbers of values, still contains over 96% of the total energy. In accordance with the Uncertainty Principle, however, some of the energy has leaked out into the fluctuation subimages. Consequently, in order to obtain an accurate approximation of $\mathbf{f}$, some of the highest energy fluctuation values—such as the ones that are visible in Figure 4.1(d)—would have to be included along with the third trend values when performing an inverse transform. This compaction of energy property, as in the 1D case, provides the foundation for the methods of compression and denoising that we shall discuss in subsequent sections.

Table 4.1 Compaction of octagon energy

Image	*Energy*	*% Total Energy*
$\mathbf{f}$	3919.06	100.00
$\mathbf{a}^1$	3830.69	97.75
$\mathbf{a}^2$	3811.06	97.22
$\mathbf{a}^3$	3777.55	96.39

4.1.3 2D wavelets and scaling images

As in the 1D case, the various levels of a wavelet transform can be computed via scalar products of the image $\mathbf{f}$ with elementary images called *scaling images* and *wavelets.* A scalar product of two images $\mathbf{f}$ and $\mathbf{g}$, both having M rows and N columns, is defined by

$$\mathbf{f} \cdot \mathbf{g} = f_{1,1}g_{1,1} + f_{1,2}g_{1,2} + \cdots + f_{N,M}g_{N,M}. \tag{4.6}$$

In other words, $\mathbf{f} \cdot \mathbf{g}$ is the sum of all the products of identically indexed values of $\mathbf{f}$ and $\mathbf{g}$.

To see how this scalar product operation relates to wavelet transforms, let's consider the 1-level horizontal fluctuation $\mathbf{h}^1$. This subimage is defined by separate calculations of trends along rows and fluctuations along columns. It follows that the values of $\mathbf{h}^1$ are computed via scalar products with wavelets obtained by multiplying values of 1D scaling signals along rows by values of 1D wavelets along columns. Each such wavelet is denoted by $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$, which is called a *tensor product* of the 1D scaling signal $\mathbf{V}_m^1$ and 1D wavelet $\mathbf{W}_n^1$.

© 2008 by Taylor & Francis Group, LLC

(For those familiar with matrix theory, we note that $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ is the product of the transpose $(\mathbf{W}_n^1)^T$, indexed from bottom up, with $\mathbf{V}_M^1$.) For instance, if we are computing a 2D Haar wavelet transform, then

$$\mathbf{V}_1^1 \otimes \mathbf{W}_1^1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1/\sqrt{2} \\ 1\sqrt{2} \end{pmatrix} \left(1/\sqrt{2}, 1/\sqrt{2}, 0, \ldots, 0\right)$$

$$= \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ -1/2 & -1/2 & 0 & \ldots & 0 & 0 \\ 1/2 & 1/2 & 0 & \ldots & 0 & 0 \end{pmatrix}.$$

Since each scaling signal $\mathbf{V}_m^1$ is a translation by $2(m-1)$ time-units of $\mathbf{V}_1^1$, and each wavelet $\mathbf{W}_n^1$ is a translation by $2(n-1)$ time-units of $\mathbf{W}_1^1$, it follows that $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ is a translation by $2(m-1)$ units along the horizontal and $2(n-1)$ units along the vertical of $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$. The values of $\mathbf{h}^1$ are computed as the scalar products $h_{m,n}^1 = \mathbf{f} \cdot (\mathbf{V}_m^1 \otimes \mathbf{W}_n^1)$.

Notice that the Haar wavelet $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$ has energy 1 and an average value of 0, as do all the other Haar wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$. Furthermore, the support of the Haar wavelet $\mathbf{V}_1^1 \otimes \mathbf{W}_1^1$ is a 2 by 2 square, and so the support of each Haar wavelet $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ is a 2 by 2 square as well. For the Daubechies wavelets, the supports of the wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$ are also small squares, although not 2 by 2 ones. The Daub4 wavelets $\mathbf{V}_m^1 \otimes \mathbf{W}_n^1$, for instance, all have supports in some 4 by 4 square (if we allow for wrap-around at image boundaries).

Similar definitions hold for the other subimages of the 1-level transform. The values of the diagonal fluctuation $\mathbf{d}^1$ are scalar products of the image with the wavelets $\mathbf{W}_m^1 \otimes \mathbf{W}_n^1$, i.e., we have $d_{m,n}^1 = \mathbf{f} \cdot (\mathbf{W}_m^1 \otimes \mathbf{W}_n^1)$. And the values of the vertical fluctuation $\mathbf{v}^1$ are scalar products with the wavelets $\mathbf{W}_m^1 \otimes \mathbf{V}_n^1$, i.e., we have $v_{m,n}^1 = \mathbf{f} \cdot (\mathbf{W}_m^1 \otimes \mathbf{V}_n^1)$. All of these wavelets have energy 1 and average value 0, and supports in small squares. The values of the trend $\mathbf{a}^1$ are scalar products of the image with the scaling images $\mathbf{V}_m^1 \otimes \mathbf{V}_n^1$, i.e., we have $a_{m,n}^1 = \mathbf{f} \cdot (\mathbf{V}_m^1 \otimes \mathbf{V}_n^1)$. Each of these scaling images has energy 1 and support in some small square.

What is true for the first level remains true for all subsequent levels. The values of each subimage $\mathbf{a}^k$, $\mathbf{h}^k$, $\mathbf{d}^k$, and $\mathbf{v}^k$ are computed by scalar products with the scaling images $\mathbf{V}_m^k \otimes \mathbf{V}_n^k$, and the wavelets $\mathbf{V}_m^k \otimes \mathbf{W}_n^k$, $\mathbf{W}_m^k \otimes \mathbf{W}_n^k$, and $\mathbf{W}_m^k \otimes \mathbf{V}_n^k$, respectively. In Figure 4.2 we show graphs of a 2-level Haar scaling image and three 2-level Haar wavelets. Notice how these images correspond with the subimages of the 2-level Haar transform. For instance, the

© 2008 by Taylor & Francis Group, LLC

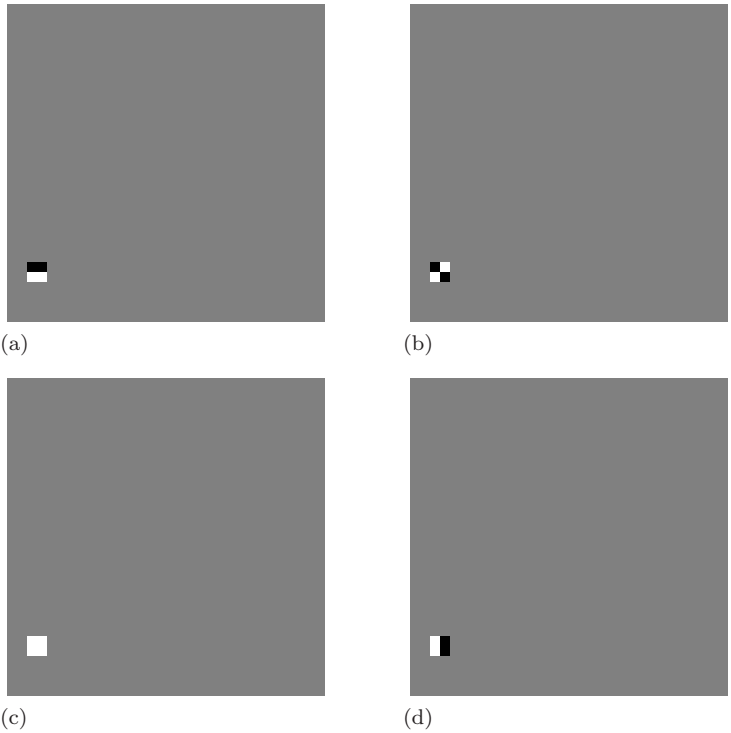


FIGURE 4.2
(a) Haar wavelet $\mathbf{V}_3^2 \otimes \mathbf{W}_5^2$. (b) Haar wavelet $\mathbf{W}_3^2 \otimes \mathbf{W}_5^2$. (c) Haar scaling image $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$. (d) Haar wavelet $\mathbf{W}_3^2 \otimes \mathbf{V}_5^2$. Note: the gray background indicates zero values, white indicates positive values, and black indicates negative values.

scaling image $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$, shown in Figure 4.2(c), is supported on a small 4 by 4 square with a constant value of 1/4, and the scalar product of an image over this square produces the trend value in the $(3, 5)$ position in the trend subimage $\mathbf{a}^2$. Also notice the horizontal orientation of the Haar wavelet in Figure 4.2(a), which is used for computing a value for the horizontal fluctuation $\mathbf{h}^2$. Similarly, the wavelets in Figures 4.2(b) and 4.2(d) have a diagonal orientation and a vertical orientation, respectively, which is consistent with their being used for computing values for the diagonal fluctuation $\mathbf{d}^2$ and vertical fluctuation $\mathbf{v}^2$.

This concludes a very brief outline of the basic features of 2D wavelets and scaling images. Their essential features are very similar to the 1D case; so for reasons of space we shall limit ourselves to this brief treatment. In the next section we shall begin our discussion of applications of 2D wavelet analysis.

4.2 Compression of images—fundamentals

One of the great success stories of wavelets is in the field of image compression.

© 2008 by Taylor & Francis Group, LLC

In the following six sections we shall examine wavelet-based image compression in some detail. Our examination will include two state of the art image compression algorithms, the ASWDR algorithm and the JPEG 2000 algorithm. The JPEG 2000 algorithm (from this point on referred to as J2K) was adopted by the International Standards Organization (ISO) as their new standard for image compression, replacing the old standard, JPEG,[3] which had been in effect since 1990 and is still used widely in digital photography and Internet photo transmission. The ASWDR algorithm is a simpler algorithm than J2K, simple enough that we can give a relatively complete description of it in the limited space that we have here. It uses a different approach than J2K, and we shall find it to be an interesting algorithm to compare with this new standard.

The images that we shall compress are discrete images that are commonly used as test images in the field of image processing. These images were chosen because they exhibit a wide variety of image properties, such as a large number of edges, different surface textures, and various combinations of light and dark shadings. They are typically 512 by 512 discrete images. Their values are gray-scale intensity values from 0 to 255—that is, quantized values at 8 bpp—where 0 indicates pure black and 255 indicates pure white, and other values indicate shades of gray between these two extremes.

Although color images are more pleasing to the eye, we focus on gray-scale images for the following three reasons.

1. Our eyes are most sensitive to changes in light intensity, rather than changes in color hues, so a standard benchmark for evaluating an image compression method is its performance on gray-scale images.

2. There are many compression applications that deal with gray-scale images, such as medical imaging (see Figure 4.4), fingerprint storage (see Section 4.3), and reconnaissance (see Figure 4.14).

3. For reasons of cost, images are generally printed in gray-scale. In fact, compressing such images for effective manuscript storage and reduced memory printing[4] are two more applications of gray-scale image compression.

Although we concentrate on gray-scale image compression, we shall briefly describe at the end of Section 4.7 how color compression is done.

In this section we provide a summary of image compression fundamentals, including a comparison of wavelet-based compression with the most widely used compression technique, the JPEG algorithm. As described previously, compression can be either lossless or lossy. Lossy compression is widely employed in digital cameras, where the default mode of storage is typically a lossy `*.jpg` file (a JPEG file). It is also the most commonly used method for

[3] JPEG stands for Joint Photographic Experts Group.
[4] A compression of the image is sent to the printer to reduce memory requirements.

© 2008 by Taylor & Francis Group, LLC

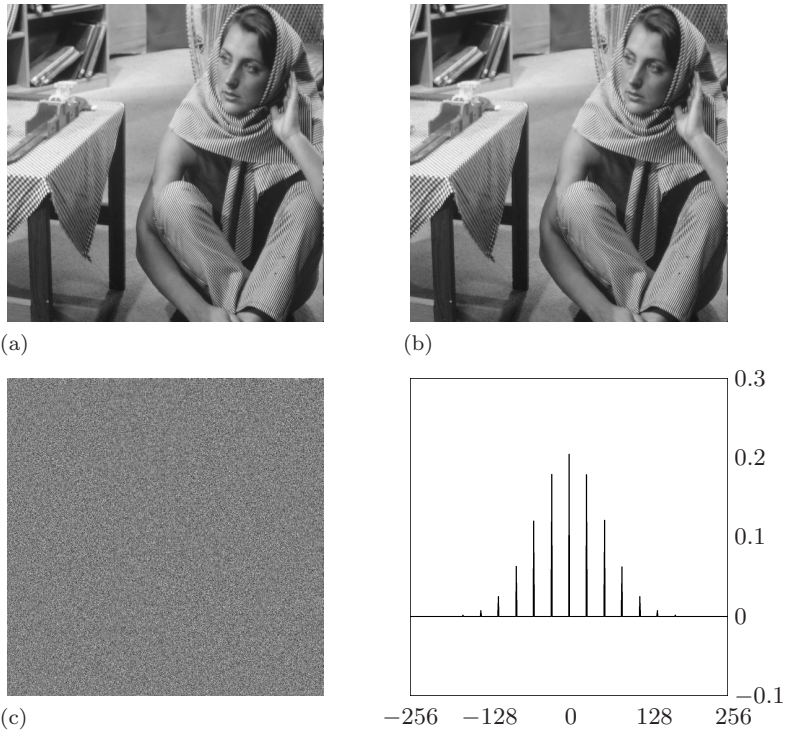


FIGURE 4.3
Perceptually Lossless Compression. (a) Original image. (b) Reconstruction of 4:1 compression. (c) Difference of images in (a) and (b). The graph to the right of image (c) is a histogram of the values in image (c).

transmitting images over the Internet (again, as `jpg` files), and in compressing photographic frames in video (the MPEG video standard,[5] for example, employs JPEG). Another application of JPEG compression is in the widely employed Portable Document Format, which uses lossy JPEG compression of images to produce compact PDF files. Because of all these practical applications, and for some fundamental theoretical reasons discussed below, we shall mostly treat lossy compression. In Section 4.6, however, we shall briefly examine lossless compression.

While lossless compression may seem to be the only acceptable paradigm, there are good reasons why lossy compression is more widely employed. For one thing, images of natural scenes are always contaminated with some random *noise,* random fluctuations of light and dark with no discernible pattern. For example, consider the *Barbara* image shown in Figure 4.3(a). In Figure 4.3(b) we show a reconstruction (decompression) of a 4:1 compression of the *Barbara* image. There is no discernible difference between the original image and its 4:1 reconstruction. In fact, in Figure 4.3(c) we show the difference

[5]MPEG stands for Motion Pictures Expert Group.

© 2008 by Taylor & Francis Group, LLC

between the two images and you can see that it consists of just random fluctuations of light and dark. The histogram in Figure 4.3 confirms the random nature of the image in Figure 4.3(c); the light and dark fluctuations in the image are distributed along a bell-shaped, normal probability distribution. In this case, we see that 4:1 compression of *Barbara* achieves *perceptually lossless* compression. Attaining true lossless compression in this case would be a futile exercise. The extra details captured by lossless compressing would only consist of random noise. No useful information would be captured.

Another example of perceptually lossless compression is shown in Figure 4.4. We see that a 16:1 compression of a medical X-ray image is perceptually indistinguishable from the original. Incidentally, both of the compressions in Figures 4.3 and 4.4 were obtained using the ASWDR algorithm that we describe in Section 4.5.

As we mentioned above, the new J2K standard is a replacement of the old standard, JPEG. We shall now briefly indicate why a new standard was needed. Although JPEG performs very well, it was desired that a new standard should enjoy at least the following three features.

1. Be able to compress much larger images than JPEG can handle.
2. Provide significantly improved compression performance. We shall see below that, *for a given level of error,* J2K can compress an image roughly twice as much as JPEG.
3. When very high compression is required, the new standard should not exhibit the annoying artifacts—known as blocking artifacts, see Figure 4.5(b)—that JPEG suffers from.

J2K achieves all of these goals. It can compress truly gargantuan images, up to 100 MB in size! It achieves the other two goals as well; we shall now briefly illustrate this. First, we must be precise about what we mean by a given level of error.

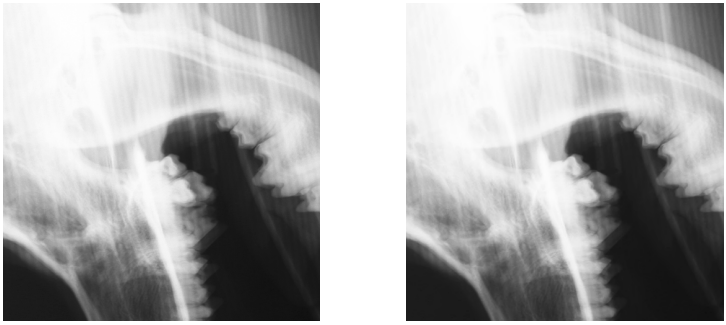
4.2.1 Error measures in image processing

There are several equivalent error measures used in image processing. We shall describe two of them. The first measure is *Mean Square Error* (MSE). Given an image $\mathbf{f}$, and an image $\mathbf{g}$ of equal dimensions, the MSE between them is defined to be

$$(\text{MSE}) = \frac{1}{P}\sum_{i,j}(f_{i,j} - g_{i,j})^2 \tag{4.7}$$

where P is the number of values of the images and the sum is over all values. This error measure is commonly employed because it is just a multiple of the energy of the difference between the images. Since the transforms used in

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 4.4
(a) X-ray image. (b) Reconstruction of its 16:1 wavelet compression.

image processing are typically energy preserving (or nearly so), this facilitates theoretical calculations involving MSE.

An equivalent measure of error, which is more commonly employed in benchmarking different algorithms in image processing, is *Peak Signal to Noise Ratio* (PSNR). Given an 8-bit gray-scale image $\mathbf{f}$, and an 8-bit gray-scale image $\mathbf{g}$ of equal dimensions, the PSNR between them is defined to be

$$(\text{PSNR}) = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right) \tag{4.8}$$

using the MSE between $\mathbf{f}$ and $\mathbf{g}$ defined in Equation (4.7). The value 255^2 in Equation (4.8) simply represents the largest possible MSE between any two 8-bit gray-scale images. Unlike MSE, a *larger* PSNR indicates a closer approximation of $\mathbf{g}$ to $\mathbf{f}$. PSNR is usually preferred in practical comparisons because it is a logarithmic measure and it has been found that our brains respond logarithmically to changes in light intensity. The units of PSNR are *decibels.* As a *rule of thumb,* if the PSNR of the original image compared to its reconstruction exceeds 40 dB, then the reconstruction will be perceptually lossless. For example, the reconstructions in Figures 4.3(b) and 4.4(b) yield PSNRs of 42.4 and 47.1 dB, respectively. For many applications, lower PSNRs are adequate. For example, in fingerprint storage a PSNR above 30 dB usually suffices (see Figure 4.7).

4.2.2 Comparing JPEG with wavelet-based compressors

The wavelet-based image compressors, J2K and ASWDR, provide significant improvements in compression performance over JPEG. In Table 4.2 we provide some PSNRs for these three methods on a few test images. Notice that both J2K and ASWDR provide, *for a given level of PSNR error,* roughly twice the compression as JPEG. This provides strong objective support for the superiority of these wavelet-based methods over JPEG, and is one of the main

© 2008 by Taylor & Francis Group, LLC

Table 4.2 PSNRs for Three Methods

Compression Ratio	*JPEG*	*J2K*	*ASWDR*
Barbara image			
8:1	32.9	35.7	36.0
16:1	27.5	32.1	31.0
32:1	24.3	28.2	27.1
Boats image			
8:1	36.4	38.4	38.8
16:1	32.1	34.4	34.1
32:1	27.9	30.8	30.6
X-ray image			
8:1	45.6	49.1	50.4
16:1	45.3	48.1	48.1
32:1	41.3	46.4	46.0

reasons that the ISO has chosen to replace JPEG with J2K as its standard for image compression.

Although PSNR values are objective, they do not completely solve the problem of specifying the best compression *to human observers.* In fact, no such measure is universally agreed on.[6] In lossy compression, we must also rely on subjective assessments of image quality, and such assessments will vary depending on the quality of reconstruction that is needed for a particular application. We shall examine this point more thoroughly in the context of fingerprint compression in the next section. To conclude the present section, we provide an illustration of the *blocking artifact* that JPEG suffers from when high compression ratios are used, and that wavelet-compression methods do not exhibit.

In Figure 4.5 we show the *Boats* image and three reconstructions of 64:1 compressions. Notice that the JPEG decompression in Figure 4.5(b) seems to consist of a collection of patchy blocks, while the wavelet decompressions in Figures 4.5(c) and (d) do not suffer from this artifact. For some applications, such as rapid Internet browsing of thumbnail images in a data bank, these wavelet-based reconstructions would be adequate. Moreover, since they are the results of 64:1 compressions, they could be transmitted at 64 times the speed of the originals over the Internet—an important feature for rapid Internet browsing, or for fast downloading of web pages with image content. Another instance of the blocking artifact is shown in the 20:1 JPEG compression of a fingerprint image in Figure 4.6(b). This example shows that the blocking artifact can occur at moderately high, not just extremely high, compression ratios. Notice also that the wavelet compressions of the fingerprint image do not exhibit this blocking artifact.

The blocking artifact in JPEG stems from the way that JPEG organizes its

[6] Although some important work has been done. See, for example, references [27], [28], and [23].

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) JPEG

(c) J2K

(d) ASWDR

FIGURE 4.5
Reconstructions of 64:1 compressions of the *Boats* image.

compression. It divides a given image into a large number of 8 by 8 subimages and separately compresses those subimages. When the compression ratio is too high, JPEG can only allocate enough bits to encode little more than the background intensity of each subimage. Because those intensities vary from subimage to subimage, one perceives a pattern of blocks. Wavelet methods use an entirely different approach, so they do not suffer from this artifact.

4.3 Fingerprint compression

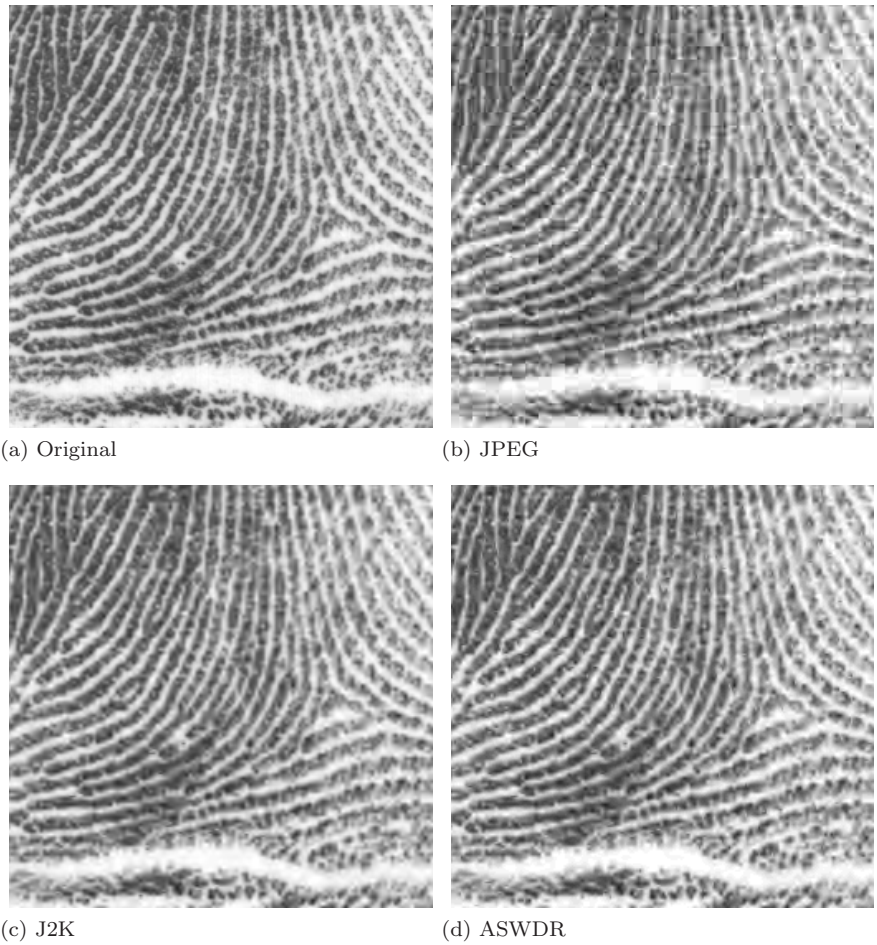
Fingerprint compression presents an interesting case study in image compression. In this section we shall briefly outline the essential ideas underlying wavelet compression of fingerprints. A wavelet-based compression algorithm has been adopted by the FBI as its standard for transmitting and storing digitized fingerprints.

A *fingerprint image* is an image of a fingerprint in an 8-bit gray-scale format. A typical fingerprint record—ten fingerprints plus two additional thumbprints and two full handprints—when digitized as images produces about 10 megabytes of data. This magnitude of data poses significant prob-

© 2008 by Taylor & Francis Group, LLC

lems for transmission and storage. For example, to transmit one fingerprint record over a modem, say at 56,000 bits per second with an overhead of around 20%, would require about half an hour. This is painfully slow when identification needs to be done quickly, as is often the case in criminal investigations. If fingerprint images could be compressed at, say, a 10:1 ratio *without noticeable loss of detail,* then the transmission of one fingerprint record could be reduced to just 3 minutes. This would greatly facilitate the large number of background checks (around 30,000 each day) that are needed.

Besides the transmission problem, there are also problems stemming from the gargantuan magnitude of the FBI's fingerprint archive. This archive contains over 25 million records. Digitizing these records at 8 bpp would produce over 250 trillion bytes of data! Compressing each image by a factor of at least



(a) Original (b) JPEG

(c) J2K (d) ASWDR

FIGURE 4.6
Zooms of reconstructions of 20:1 compressions of fingerprint image.

© 2008 by Taylor & Francis Group, LLC

10:1 would greatly ease the storage burden for an archive of these fingerprint images.

Initially the FBI tried to use the JPEG algorithm for compressing fingerprint images. But they were not pleased by the blocking artifacts that showed up. An example is shown in Figure 4.6(b). Notice that those blocking artifacts obscure some of the fine details of the image, such as the round white dots within the fingerprint ridges. The FBI felt that these artifacts significantly degraded the reconstructions, making them unsuitable for some identification procedures. Hence they began a search for a new compression algorithm.

The compression algorithm adopted by the FBI is called the *Wavelet Scalar Quantization* (WSQ) method. We will not try to give a complete description of the WSQ method. Rather, we shall describe enough of its essential aspects that the reader should be able to confidently read some of the excellent articles by its creators. These articles are listed in the Notes and References section at the end of this chapter.

We shall now illustrate the gist of the WSQ method by discussing an example of a fingerprint compression using a wavelet transform. This initial example will be expanded in Chapter 6, in order to give further insight into

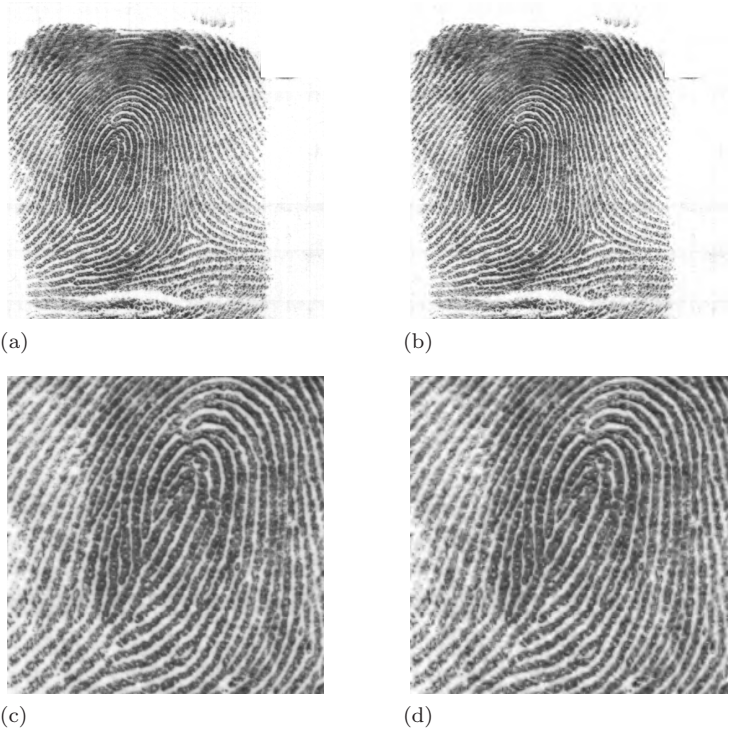


FIGURE 4.7
(a) Fingerprint 1 image, 8 bpp. (b) Compressed version, 0.8 bpp (PSNR = 32.7 dB). (c) Central whorl in (a). (d) Central whorl in (b).

© 2008 by Taylor & Francis Group, LLC

the WSQ method.

Consider the test image, `Fingerprint 1.bmp`, shown in Figure 4.7(a). To compress this image we used a 7-level Daub 9/7 transform at a rate of 0.8 bpp, a 10:1 compression. The method used was the ASWDR method that we describe in Section 4.5. The reconstruction of this compression is shown in Figure 4.7(b). This image is virtually indistinguishable from the original image in (a). As a quantitative measure of the accuracy of the compressed image, we calculated the PSNR defined in Equation (4.8). We found a PSNR of 32.7 dB. This value is higher than the rule-of-thumb threshold of 30 dB that we mentioned in the previous section.

Often portions of fingerprints need to be magnified in order to compare certain details, such as *whorls*. In Figures 4.7(c) and (d) we show magnifications of the central whorl from `Fingerprint 1.bmp` and from its compressed version. The compressed version is still virtually indistinguishable from the original in these magnifications. It is interesting to note that even the tiny indentations in the middle of the fingerprint ridges, which correspond to sweat pores, are accurately preserved by the compressed image. The locations of these sweat pores are, in fact, legally admissible for identification purposes.

While this example illustrates that a wavelet transform compression of fingerprints can work reasonably well, we can do even better. In Chapter 6, we will discuss how a modified transform, the one employed by the FBI's WSQ method, leads to improved compression.

4.4 The WDR algorithm

In the next section, we describe the ASWDR algorithm that we illustrated in the previous two sections. The ASWDR algorithm is an improvement of an earlier algorithm called *Wavelet Difference Reduction* (WDR). In this section we describe the WDR algorithm, which is also interesting in its own right. We begin by describing *bit-plane encoding* using wavelet difference reduction. Bit-plane encoding elegantly achieves the savings in bit-rate of two-threshold quantization that we illustrated in the previous chapter for 1D compression. That is just one of its advantages. It also allows for precise specification of a desired compression ratio (something that not all image compression algorithms allow for). We conclude the section by describing a method of *arithmetic compression.* WDR uses arithmetic compression as an additional component on top of its basic component of bit-plane encoding.

4.4.1 Bit-plane encoding

All of our compression algorithms perform a wavelet transform of a gray-scale image. For lossy compression a Daub 9/7 transform is used, and for lossless compression an integer-to-integer Daub 5/3 transform is used. Following the wavelet transform, these algorithms calculate a bit-plane encoding of the transform values (although J2K includes some refinements to maximize

© 2008 by Taylor & Francis Group, LLC

PSNR). The method of bit-plane encoding essentially just performs binary expansions of wavelet transform values relative to some initial threshold value. The three steps of the bit-plane encoding procedure are as follows.

Bit-plane Encoding Procedure

Step 1. INITIALIZE. Choose initial threshold T_0 so that all transform values are smaller in magnitude than T_0 but at least one transform value is greater than or equal to $T_0/2$. Transmit this threshold value, along with information on the dimensions of the image, which wavelet transform was used, and how many levels of transform were computed.

Step 2. SIGNIFICANCE PASS. Set threshold $T_k = T_{k-1}/2$. Encode signs and locations (indices) of transform values w that satisfy

$$T_k \le |w| < T_{k-1}.$$

We shall explain below how this encoding is done. Transmit or store this encoded information, which specifies the *quantized value* $w_Q = \text{sign}(w)\, T_k$ in place of the value w at its location within the wavelet transform. After encoding each value, check to see if the specified compression rate is attained. If it is then end the procedure.

Step 3. REFINEMENT PASS. Examine every transform value w that satisfies $|w| \ge T_{k-1}$ with a goal of obtaining a quantized value w_Q of accuracy within T_k. If $|w| \ge |w_Q| + T_k$, then w_Q should be refined to the new value $w_Q = (w_Q)_{\text{old}} + \text{sign}(w)\, T_k$. On the other hand, if $|w| < |w_Q| + T_k$, then w_Q should retain its present value. Whether to refine or not is indicated by transmitting or saving either the bit 1, when refinement is needed, or the bit 0, when refinement is not needed. After outputting a refinement bit for each such transform value, check to see if the specified compression ratio has been attained. If it has, then end the procedure. If this step is completed for all transform values, then loop back to Step 2.

For example, consider the wavelet transform shown in Figure 4.8(b). The values of this transform are scanned through according to the numbering scheme shown in Figure 4.8(a) (more on this below). The initial threshold for this wavelet transform is $T_0 = 32$. In Figure 4.8(c) we show the quantized wavelet transform values after completing the Significance Pass with $T_1 = 16$ (notice that for T_1 the Refinement Pass is always skipped). In Figure 4.8(d) we show the quantized wavelet transform values after completing the Significance Pass and the Refinement Pass with $T_2 = 8$. Finally, in Figure 4.9(a) we show the quantized wavelet transform values after completing the Significance Pass and the Refinement Pass with $T_3 = 4$. Notice how we are successively obtaining more accurate approximations to the original wavelet transform. In fact, because we started with an integer-valued wavelet transform, if we were

© 2008 by Taylor & Francis Group, LLC

32	31	30	29	55	61	62	64
25	26	27	28	54	56	60	63
24	23	22	21	50	53	57	59
17	18	19	20	49	51	52	58
8	7	14	16	36	37	44	45
5	6	13	15	35	38	43	46
2	4	9	12	34	39	42	47
1	3	10	11	33	40	41	48

(a) Scan order

4	6	8	6	2	4	2	6
−2	−3	−4	−2	1	3	−2	−4
−4	−5	−6	−5	3	−2	−6	−8
3	4	6	5	2	3	−1	−2
7	4	6	8	10	−5	4	7
9	10	7	4	9	−7	6	8
14	28	−10	8	9	−6	5	10
22	24	−9	11	8	−5	8	9

(b) Wavelet transform

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	16	0	0	0	0	0	0
16	16	0	0	0	0	0	0

(c) Threshold = 16

0	0	8	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	−8
0	0	0	0	0	0	0	0
0	0	0	8	8	0	0	0
8	8	0	0	8	0	0	8
8	24	−8	8	8	0	0	8
16	24	−8	8	8	0	8	8

(d) Threshold = 8

FIGURE 4.8
First two stages of bit-plane encoding. (a) 2-level scan order. (b) 2-level wavelet transform. (c) Stage 1, threshold = 16. (d) Stage 2, threshold = 8.

to complete both Passes for a threshold of 1, then we would obtain the exact wavelet transform and our compression would be lossless. When the wavelet transform has floating point (decimal) values, then we generally would not be able to produce an exact copy of the original transform because of the constraint of the compression ratio. Our compression would then be lossy.

When decompressing, the above steps are recapitulated based on the information transmitted or stored during encoding, and an approximate wavelet transform is constructed. When lossy compression has been performed, a further approximation of the original wavelet transform is carried out by adding half of the last threshold value to the magnitudes of the quantized values. For instance, in Figure 4.9(b) we show how this last approximation is done for

© 2008 by Taylor & Francis Group, LLC

4	4	8	4	0	4	0	4
0	0	−4	0	0	0	0	−4
−4	−4	−4	−4	0	0	−4	−8
0	4	4	4	0	0	0	0
4	4	4	8	8	−4	4	4
8	8	4	4	8	−4	4	8
12	28	−8	8	8	−4	4	8
20	24	−8	8	8	−4	8	8

(a)

6	6	10	6	0	6	0	6
0	0	−6	0	0	0	0	−6
−6	−6	−6	−6	0	0	−6	−10
0	6	6	6	0	0	0	0
6	6	6	10	10	−6	6	6
10	10	6	6	10	−6	6	10
14	30	−10	10	10	−6	6	10
22	26	−10	10	10	−6	10	10

(b)

FIGURE 4.9
(a) Stage 3, Threshold = 4: MSE = 2.98437. (b) Last stage of reconstruction, half-threshold accuracy: MSE = 2.34375 (a 23.86% reduction in MSE).

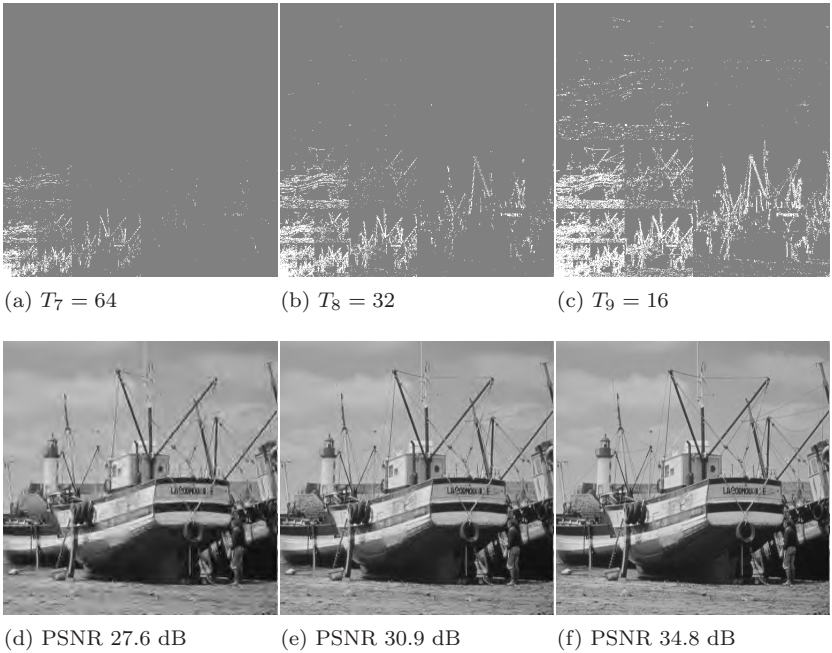
the example considered above. Notice how this reduces the MSE between the quantized transform and the original transform by about 24%. This reduction of MSE is typical and is the reason this extra step is performed.

An example illustrating bit-plane encoding as it relates to an actual image is shown in Figure 4.10. This figure merits close examination. It is typical of how bit-plane encoding works on images of natural scenes. As the thresholds decrease, more and more values are captured by the bit-plane encoding, which adds more and more details to the reconstructed images, thus producing increasingly sharper and more accurate reconstructions. Notice especially that the level of where values are captured by the bit-plane encoding corresponds to the magnitude of the threshold, i.e., the higher thresholds capture more values at the higher levels in the transform, and the lower thresholds capture more details at the lower levels in the transform.

4.4.2 Difference reduction

We now turn to the problem of how the positions of significant values are encoded during the Significance Pass. Our discussion will also make clear how the decoder will be able to correctly interpret the sequence of bits generated by the Refinement Pass. As we pointed out above, during the Significance Pass and during the Refinement Pass, the values of the wavelet transform are scanned through in a particular order. An example of such a scanning order is shown in Figure 4.8(a). The pattern of the numbering scheme for this scanning order is the following. First, for a wavelet transform of L levels, the trend subimage $\mathbf{a}_L$ is scanned through in a *zig-zag* pattern. For instance, if $\mathbf{a}_L$ was a 3 by 3 subimage, rather than the 2 by 2 one in Figure 4.8(a), then

© 2008 by Taylor & Francis Group, LLC



(a) $T_7 = 64$ (b) $T_8 = 32$ (c) $T_9 = 16$

(d) PSNR 27.6 dB (e) PSNR 30.9 dB (f) PSNR 34.8 dB

FIGURE 4.10
Illustration of Bit-Plane Encoding of Boats Image [Original in Figure 4.5(a)]. Images on top show white pixels for locations of significant values (magnitudes greater than or equal to threshold) for a 5-level Daub 9/7 transform. Images on the bottom are reconstructions for the thresholds indicated above them.

the scanning order through the values of $\mathbf{a}_L$ would be as shown in Table 4.3.

Second, the subimages of the wavelet transform are scanned through in the following order (ordered according to the degree of compaction of energy within the subimages of a wavelet transform):

$$\mathbf{a}_L, \mathbf{h}_L, \mathbf{v}_L, \mathbf{d}_L, \mathbf{h}_{L-1}, \mathbf{v}_{L-1}, \mathbf{d}_{L-1}, \ldots, \mathbf{h}_1, \mathbf{v}_1, \mathbf{d}_1.$$

Third, the scanning through each horizontal subimage $\mathbf{h}_k$ is done in a *row-wise* manner, the scanning through each vertical subimage is done in a *column-wise* manner, and the scanning through each diagonal subimage $\mathbf{d}_k$ is done in a zig-zag manner. See Figure 4.8(a), where the scanning methods are most clearly illustrated for the 4 by 4 subimages $\mathbf{h}_1$, $\mathbf{v}_1$, $\mathbf{d}_1$.

Given a scanning order through the wavelet transform values, the position of a significant value can be specified as indicated in the following example.

Table 4.3 An $\mathbf{a}_L$ scan order

6	7	9
2	5	8
1	3	4

© 2008 by Taylor & Francis Group, LLC

Suppose that there are significant values at positions 3, 7, and 17, respectively. These positions are then specified by the numbers 3, 4, and 10. The values of 4 and 10 indicate the *number of steps* through the scan order needed to get from the previous value to the present one. We can reconstruct the position 7 by adding 4 to 3, and then reconstruct 17 by adding 10 to 7. We also have to transmit information that indicates that the significance pass is complete. We do that by transmitting a plus sign and the number of steps to the last position in the scan order plus 1; those number of steps would then take us beyond the last position in the scan order thus indicating that the significance pass is completed. Suppose, for example, that those number of steps is 37. These numbers 3, 4, 10, and 37 are then expressed in binary notation as

$$3 = (1\ 1)_2, \quad 4 = (1\ 0\ 0)_2, \quad 10 = (1\ 0\ 1\ 0)_2, \quad 37 = (1\ 0\ 0\ 1\ 0\ 1)_2.$$

It is important to note that when we make our step counts, *we skip over previously encoded significant values,* thus reducing the lengths of these binary expansions. Notice also that the leading bit in these binary expressions is always a 1. Therefore, 1 does not need to be transmitted or stored. Instead, it can be replaced by the sign of the significant value whose position is being specified (a plus sign is always used for the end marking number). Those signs serve as separators between the different encodings of the number of steps. Finally, at the end of the significance pass, a plus sign is attached to mark the end of the bits for the last binary expansion.

For the refinement pass, we simply observe that the bits indicating whether refinement is to be done are always sent in the order specified by the scan order. Hence they can just be sent in a stream of 0's and 1's.

Let's see how this works for the wavelet transform in Figure 4.8(b) with the scan order shown in Figure 4.8(a). For the threshold $T_1 = 16$ there are significant values of $+22$, $+24$, and $+28$ at the positions 1, 3, and 4 respectively. The number of steps are then $1 = (1)_2$, $2 = (1\ 0)_2$, and $1 = (1)_2$. There are 60 steps to the end of the scan order; thus we will encode $61 = (1\ 1\ 1\ 1\ 0\ 1)_2$ for this case. Our encoding at the completion of the significance pass is therefore

$$++0++11101+ \tag{4.9}$$

Since this is the first threshold, there are no refinement bits to transmit. For the threshold $T_2 = 8$, there are a large number of *new* significant values. Counting through the scan order, remembering to skip over old significant values, we obtain the following counts:

$$1, 1, 1, 3, 1, 1, 1, 4, 14, 3, 1, 1, 1, 5, 5, 1, 1, 11, 6.$$

Thus our encoding at the end of the second significance pass is

$$+++-1-++++00+110+1++++01+01++-011+10+ \tag{4.10}$$

At the end of the refinement pass we would have encoded the following bits

011

© 2008 by Taylor & Francis Group, LLC

indicating whether refinements are needed for the three old significant values.

When transmitting or storing this compressed data, an *arithmetic compression* is usually performed to use as few bits as possible.

4.4.3 Arithmetic compression

All computer files, whether transmitted or stored, are ultimately just sequences of the two bits 0 and 1. If the symbols +, −, 0, and 1 in WDR encoding occurred with equal frequencies, then the shortest binary encoding of those symbols would be two bits for each symbol. Here is a table for such an encoding:

Symbol	*Bits*
+	00
−	01
0	10
1	11

(4.11)

The fact is, however, these symbols do not occur with equal frequency. For instance, look at the two sequences in (4.9) and (4.10). For those sequences, the symbols occur with the frequencies shown in the second column of Table 4.4.

Table 4.4 Data from (4.9) and (4.10)

Symbol	*Frequency*	*Average bits for encoding*
+	22/47	$\log_2(47/22)$
−	3/47	$\log_2(47/3)$
0	9/47	$\log_2(47/9)$
1	13/47	$\log_2(47/13)$

The idea of arithmetic compression is to assign *variable length* bit sequences to such symbols of *average* length $\log_2(1/p)$ where p is the frequency (probability) of occurrence of a given symbol. Hence, for this example, these variable length bit sequences would be assigned in such a way that the average length of the bit sequences for each symbol would be those given in the third column of Table 4.4. Thus we find that the total number of bits needed for encoding the sequences of symbols in (4.9) and (4.10) to be

$$22\log_2(47/22) + 3\log_2(47/3) + 9\log_2(47/9) + 13\log_2(47/13) \approx 82 \text{ bits.}$$

This compares favorably with the total of 94 bits needed if the encoding using two bits per symbol in (4.11) were used.

How does arithmetic coding work? We will explain the gist of the method. Details of the actual computer implementation are left to the excellent discussions given in the references for this chapter.

To begin, observe that the two-bit code in (4.11) could be thought of as corresponding to a uniform partitioning of an interval, say $(0, 1)$, into subin-

© 2008 by Taylor & Francis Group, LLC

tervals using the following pairings:

$$\begin{aligned} + &\leftrightarrow (0, 1/4) \\ - &\leftrightarrow (1/4, 1/2) \\ 0 &\leftrightarrow (1/2, 3/4) \\ 1 &\leftrightarrow (3/4, 1). \end{aligned}$$

These intervals are all of equal length $1/4$, matching the equal frequencies for each bit, and $\log_2[1/(1/4)] = 2$, the number of bits needed for each symbol. Suppose, for instance, that the first symbol was a $+$. Then the interval $(0, 1/4)$ would be specified. For encoding the next symbol, $(0, 1/4)$ would be divided into 4 equal length subintervals by the same scheme. For instance, if the second symbol was a 0, then the subinterval $(1/8, 3/16)$ of length $1/16$ would be specified. If the third symbol was a $-$, then a subinterval of *relative position* $(1/4, 1/2)$ within $(1/8, 3/16)$ would be specified, i.e.,

$$\left(\frac{1}{8} + \frac{1}{4} \cdot \frac{1}{16}, \frac{1}{8} + \frac{1}{2} \cdot \frac{1}{16}\right) = \left(\frac{9}{64}, \frac{5}{32}\right).$$

Encoding a sequence of symbols with our two-bit encoding *corresponds to specifying a number within* $(0, 1)$*, in this case the number* $19/128$ *within the subinterval* $(9/64, 5/32)$*, using an iterated sequence of* 4 *subintervals, each of relative length* $1/4$. The rule for choosing $19/128$ is to select the dyadic fraction $(2k+1)/2^n$ of odd numerator, and least power $n > 1$, that lies closest to the left endpoint of the subinterval. Working out the binary expansion of $19/128$, we get

$$\frac{19}{128} = \frac{0}{2} + \frac{0}{4} + \frac{1}{8} + \frac{0}{16} + \frac{0}{32} + \frac{1}{64} + \frac{1}{128}.$$

Notice that the numerators of the fractions on the right of the last equation are $0\,0\,1\,0\,0\,1\,1$ which (ignoring the last 1, which is always 1) yields the encoding bits for $+\,0\,-$.

This correspondence between intervals, binary expansions, and encodings of symbols works for non-uniform frequencies as well. Suppose, for example, that the frequencies for our symbols satisfy

Symbol	*Frequency*	*Bit-length*
$+$	$1/2$	$\log_2[1/(1/2)] = 1$
$-$	$1/4$	$\log_2[1/(1/4)] = 2$
0	$1/8$	$\log_2[1/(1/8)] = 3$
1	$1/8$	$\log_2[1/(1/8)] = 3$

(4.12)

A variable length encoding of such symbols and their associated intervals is

© 2008 by Taylor & Francis Group, LLC

given by

Symbol	*Code*	*Interval*
+	0	$(0, 1/2)$
−	1 0	$(1/2, 3/4)$
0	1 1 0	$(3/4, 7/8)$
1	1 1 1	$(7/8, 1)$

(4.13)

Notice that the number of bits used for a given interval is inversely related to the length of the interval. Less precision (longer interval) means less information is specified, hence less bits. Moreover, *encoding a sequence of these symbols corresponds to locating a number (specifying a subinterval) within* $(0, 1)$ *with a finite iteration of division by coding intervals of relative lengths* $1/2$, $1/4$, $1/8$, $1/8$. For example, the reader may wish to verify that the sequence $+\ -$ corresponds to the subinterval $(1/4, 3/8)$, and that $+\ -\ 1$ corresponds to $(23/64, 3/8)$. These subintervals enclose the points $5/16$ and $47/128$, respectively, which have binary expansions

$$\frac{5}{16} = \frac{0}{2} + \frac{1}{4} + \frac{0}{8} + \frac{1}{16}$$
$$\frac{47}{128} = \frac{0}{2} + \frac{1}{4} + \frac{0}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128}.$$

In each case, we see that (ignoring the last numerators which are always 1) the numerators in these expansions are 0 1 0 and 0 1 0 1 1 1. Those numerators are exactly the codes for the symbol sequences! This is no accident. It is precisely how arithmetic coding works.

Note that for this variable length coding, the average number of bits needed for a symbol is

$$\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1.75,$$

which is 12.5% smaller than the 2 bits needed per symbol if the two-bit code in (4.11) were used. Here we see that a variable length code achieves more compression.

Finally, we briefly describe how arithmetic coding can achieve *fractional* average bits per symbol, such as those given in the third column of Table 4.4. To see how, let's look at a simpler case. Suppose that we have only two symbols, a and b, with frequencies 0.9 and 0.1, respectively. The coding intervals for these two symbols are then $(0, 0.9)$ for a, and $(0.9, 1)$ for b. Suppose we wish to encode the 7-symbol sequence

$$a\, a\, a\, a\, a\, a\, b.$$

The first six a's correspond to the interval $(0, (0.9)^6) = (0, 0.531441)$. The seventh symbol, b, corresponds to an interval of relative position $(0.9, 1)$ within the interval $(0, 0.531441)$, i.e.,

$$(0.531441 * 0.9, 0.531441) = (0.4782969, 0.531441).$$

© 2008 by Taylor & Francis Group, LLC

Now, observe that the number $17/32 = 0.53125$ lies within this last interval.[7] The binary expansion of this number is

$$\frac{17}{32} = \frac{1}{2} + \frac{0}{4} + \frac{0}{8} + \frac{0}{16} + \frac{1}{32}\,.$$

Hence, sending the four bits 1 0 0 0 (the fifth bit of 1 being assumed) would specify the number 0.53125. The reader should check that the shortest symbol string whose coding interval would definitely require sending the binary expansion of 0.53125 is our sequence $a\,a\,a\,a\,a\,a\,b$. Thus we have encoded 7 symbols with 4 bits, and that yields fractional average bits per symbol. Notice also that the theoretical lower bound on the number of bits needed to encode our sequence is

$$6 \cdot \log_2(1/0.9) + 1 \cdot \log_2(1/0.1) = 4.2339$$

which rounds to 4 bits. Thus, we have attained the minimum length encoding of our symbol sequence.[8] As we pointed out above, the practical details of how this arithmetic is carried out in actual computer code is described clearly in the references (see especially the book by Bell, Cleary, and Witten).

With all of these examples, for any given symbol of frequency p, the length of a coding interval is p, and $\log_2(1/p)$ is the average number of bits needed. This is true of all arithmetic codes. The method of arithmetic coding is a technique for performing such an encoding with any finite sequence of symbols $\{x_k\}_{k=1}^N$ and their associated frequencies $\{p_k\}_{k=1}^N$. Encoding sequences of such symbols corresponds to locating numbers with finite precision within $(0, 1)$ using a nested hierarchy of subintervals of *relative* lengths $\{p_k\}$, via the following scheme:

Symbol	*Interval*
x_1	$(0, p_1)$
x_2	$(p_1, p_1 + p_2)$
$\vdots$	$\vdots$
x_N	$(p_1 + p_2 + \cdots + p_{N-1}, 1)$

The average coding length, bits per symbol, of such an arithmetic code is then

$$p_1 \log_2(1/p_1) + p_2 \log_2(1/p_2) + \cdots + p_N \log_2(1/p_N), \tag{4.14}$$

which is the minimum coding length (entropy) defined (mutatis mutandis) in Equation (3.37) on p. 67.

[7] The number 1/2 also lies in the interval, but the single bit 1 is never transmitted, as it corresponds to the whole interval $(0, 1)$.

[8] The theoretical lower bound applies to *averages* over the number of bits needed for encoding many sequences, so it can be fractional and a *particular* sequence can occasionally be encoded with fewer bits.

© 2008 by Taylor & Francis Group, LLC

It is important to note that arithmetic coding applies to more than just the situation of single symbols viewed in isolation, and their associated frequencies. In fact, it constitutes the foundation for an entire paradigm of statistical modelling of a source of symbols and optimal compression of sequences generated from such a source. We shall discuss this further in the next section.

We now finish our discussion of the WDR algorithm. By performing arithmetic compression on the symbols $+, -, 0, 1$ of WDR's output, its compression performance is definitely improved. In Table 4.5 we show PSNRs for several compression ratios on some test images. The column headed *WDR (No AC)* gives the PSNRs when the 2-bit code in (4.11) is used without any arithmetic compression. The column headed *WDR* shows the PSNRs when arithmetic compression is applied. It is clear that arithmetic compression does increase the PSNRs for these compressions, by an average of about 0.18 dB. We shall see in the next section that arithmetic compression provides slightly more improvement for the adaptively scanned version of WDR, the ASWDR algorithm.

Table 4.5 PSNR Values for Four Methods

Comp. Ratio	*WDR (No AC)*	*WDR*	*ASWDR (No AC)*	*ASWDR*
		Barbara image		
8:1	35.3	35.5	35.7	36.0
16:1	30.5	30.7	30.8	31.0
32:1	26.7	26.9	27.0	27.1
		Boats image		
8:1	38.2	38.4	38.4	38.8
16:1	33.5	33.8	33.8	34.1
32:1	30.2	30.4	30.4	30.6
		X-ray image		
8:1	50.1	50.2	50.2	50.4
16:1	47.7	47.8	47.8	48.1
32:1	45.5	45.6	45.8	46.0

4.5 The ASWDR algorithm

The ASWDR algorithm adds a single new feature to WDR, *adaptive scanning.* At the end of a Refinement Pass, ASWDR calculates a new scanning order through the wavelet transform values. The purpose of creating a new scanning order is to reduce the number of steps between new significant values, thus reducing the symbol output within the next Significance Pass and thereby increasing compression.

We now describe the rationale underlying the creation of a new scanning order. The main principle is the following:

© 2008 by Taylor & Francis Group, LLC

Principle A. *An old significant value at position (i, j) in a level m fluctuation subimage (called a* parent*) indicates that at least one of the fluctuation values at $(2i, 2j)$, $(2i + 1, 2j)$, $(2i, 2j + 1)$, $(2i + 1, 2j + 1)$ in level $m - 1$ (called its* children*) is probably significant at the new threshold value.*

For example, consider Figure 4.11. In Figure 4.11(a) we show child positions (not previously encoded as significant) in the fluctuation subimage $\mathbf{v}^1$ corresponding to old significant parent positions in the fluctuation subimage $\mathbf{v}^2$. These provide very good predictors of the new significant values in $\mathbf{v}^1$ shown in Figure 4.11(b). In fact, we get 41.2% correct predictions of new significant values. Figure 4.11(c) and (d) shows similar predictions and actual locations of new significant values, with 79.7% correct predictions. Scanning through these predicted positions first, we will reduce the number of steps needed between successive new significant values, and thereby reduce the symbol output in the significance pass. Hence, the first part of a new scanning order at level $m-1$ is a scanning through (not previously encoded) children of old significant parents at level m.

The reason that Principle A holds is that the support of a wavelet at level m corresponding to position (i, j) contains the supports of the four wavelets at level $m-1$ corresponding to the four positions $(2i, 2j)$, $(2i+1, 2j)$, $(2i, 2j+1)$, $(2i + 1, 2j + 1)$. A significant value at position (i, j) at level m will typically occur when the support of a corresponding wavelet overlaps an edge in the image. In which case, one or more of the supports of the wavelets corresponding to the positions of its children at level $m - 1$ will also overlap the edge. For example, in Figure 4.11, one can see the correlation between significant positions and edges in the *Boats* image.

But what if a parent value is *insignificant,* while one or more of the children is significant? Such values are not predicted as newly significant by Principle A. ASWDR makes a second prediction for these values based on the following principle:

Principle B. *If a child value is significant at an old threshold value, then it is likely that another child value (one of its* siblings*) will be significant at the new threshold value.*

Relying on Principle B, ASWDR creates a second part of the scan order, consisting at a given level of those siblings (that were not previously encoded) of old significant child values whose parents are insignificant at older thresholds.

The rationale for Principle B is that the supports of the four children greatly overlap each other. So when a support of a wavelet overlaps an edge it is quite likely that one of its sibling's supports also overlaps the edge. For example, in Figure 4.11(b) and (d) we can see that new significant values are clustered together about the outlines of the edges in the boats. They are frequently clustered together as adjacent siblings. This accounts for the vast majority of the new significant values not captured by the first part of the new scan order.

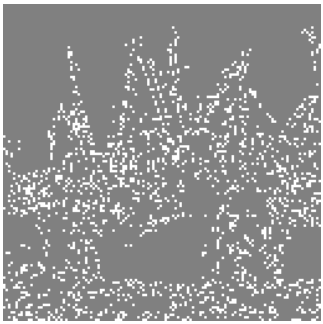
© 2008 by Taylor & Francis Group, LLC

Finally, the new scan order concludes by scanning through at each level the remaining insignificant values (child values that have insignificant parents and all of whose siblings are also insignificant). For example, in Figures 4.11(b) and (d), those values correspond to the large gray regions.

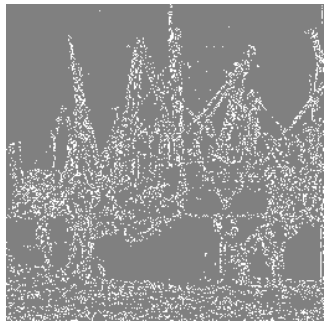
Using this new scan order along with the 2-bit encoding in (4.11), we obtain the PSNRs shown in the column labeled *ASWDR (No AC)* in Table 4.5. Even without arithmetic compression, ASWDR is able to perform slightly better than WDR with arithmetic compression. To obtain even higher PSNRs, we apply arithmetic compression to the symbols output by ASWDR.

4.5.1 Arithmetic compression

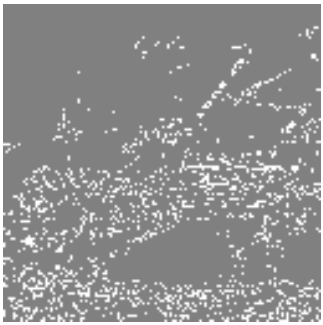
The arithmetic compression employed in ASWDR is an enhanced version of the method described in the previous section. Because there are an increased number of symbols with repetitions of signs (due to steps of length 1 be-



(a) $\mathbf{v}^2$ predictions for $\mathbf{v}^1$



(b) $\mathbf{v}^1$ new significant values



(c) $\mathbf{h}^3$ predictions for $\mathbf{h}^2$

(d) $\mathbf{h}^2$ new significant values

FIGURE 4.11
Illustration of Parent-Child significance correlation for Boats image. (a) and (c) are predictions of locations (white pixels) of new significant values based on parent values significant at threshold 16. (b) and (d) are actual locations (white pixels) of *new* significant values at child positions for threshold 8. For (a) and (b), 41.2% of the predictions were correct. For (c) and (d), 79.7% of the predictions were correct.

© 2008 by Taylor & Francis Group, LLC

tween new significant values) we have found that arithmetic compression is enhanced by setting up four separate copies, called *contexts*, of the interval $(0, 1)$ depending on whether the last encoded symbol was a $+$, $-$, 0, or 1, and splitting the frequency counts accordingly.[9] For example, for the symbol sequences in (4.9) and (4.10) we obtain the following frequency counts for these four contexts (the initial context is assumed to be $+$):

+ Context		− Context		0 Context		1 Context	
Symb.	*Freq.*	*Symb.*	*Freq.*	*Symb.*	*Freq.*	*Symb.*	*Freq.*
+	12/22	+	1/3	+	4/9	+	5/13
−	2/22	−	0/3	−	0/9	−	1/13
0	4/22	0	0/3	0	1/9	0	4/13
1	4/22	1	2/3	1	4/9	1	3/13

With those frequencies, arithmetic encoding needs

$$12\log_2(22/12) + 2\log_2(22/2) + \cdots + 3\log_2(13/3) \approx 78 \text{ bits}$$

for encoding (where a symbol of frequency 0 is not included in the sum on the left). This is an improvement over the 82 bits that we discussed in the previous section. It also indicates that this method of arithmetic coding would work better for WDR as well, and we have found this to be so. In fact, the entries in Table 4.5 for WDR were computed using this improved arithmetic coding.

In the last column of Table 4.5 we give the PSNRs for ASWDR using this improved arithmetic coding. In every case there is an increase in PSNR, between 0.2 and 0.3 dB.

4.5.2 Relation to vision

Unlike J2K—which we shall see attaches the highest priority to encoding new values that provide the greatest increase in PSNR—the ASWDR algorithm attaches the highest priority at any level to encoding new significant values that lie along edges. There is some evidence that the visual cortex within our brains decomposes an image into edges oriented either horizontally, vertically, or diagonally at four different levels of scale. So, in a loose sense, ASWDR is related to the nature of how images are decomposed by our visual system. That provides some explanation of why ASWDR can produce reconstructions that appear sharper visually than J2K. *By attaching highest priority to encoding new significant values along edges, ASWDR is attaching the highest priority to refining the details of the edges of images, thus sharpening the appearance of images.* For example, in Figures 4.12 and 4.13, we show two portions of 32:1 reconstructions of the *Barbara* image using WDR, ASWDR, and J2K. Notice that ASWDR generally preserves the most details, especially

[9] In the field of arithmetic coding, these contexts are referred to as Markov-1 states. J2K uses a much more sophisticated set of contexts.

© 2008 by Taylor & Francis Group, LLC

(a) Portion of Original (b) WDR

(c) J2K (d) ASWDR

FIGURE 4.12
Comparison of 32 to 1 compressions of *Barbara* image.

the eyes and mouth of Barbara in Figure 4.12 and the stripes of the tablecloth on the table top in Figure 4.13.

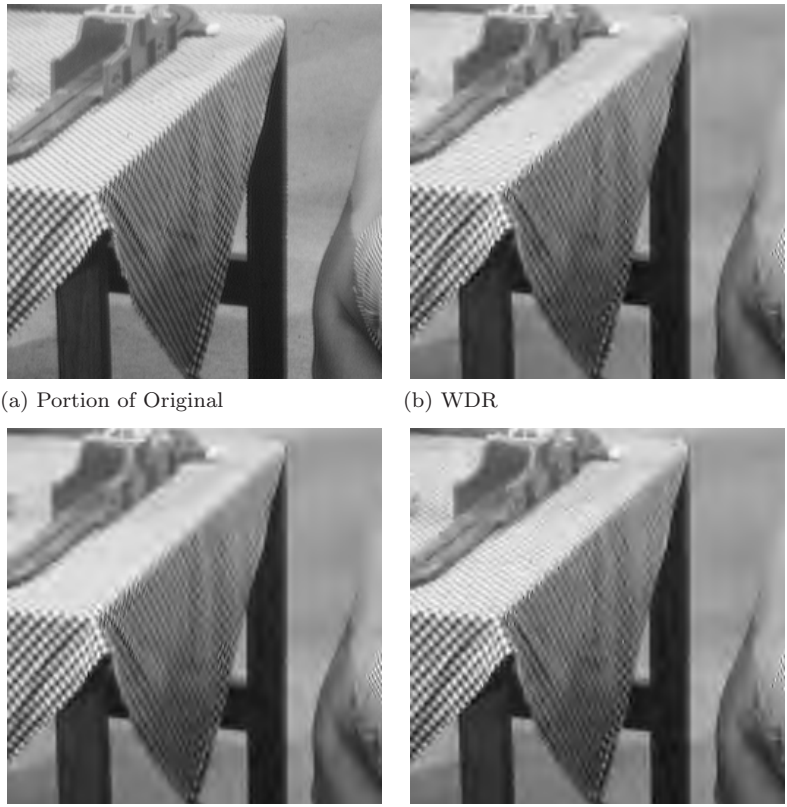
4.6 Important image compression features

All of the image compression algorithms discussed here, WDR, ASWDR, and J2K, share some important features that make them powerful tools for applications requiring image compression. These features include the following: (1) Progressive transmission/reconstruction; (2) Lossless compression option; (3) Region-of-Interest option. We shall now discuss each of these features for the case of ASWDR.

4.6.1 Progressive transmission/reconstruction

In the bit-plane encoding procedure described in Section 4.4, we pointed out that the encoder can exit the procedure at any point. It does not have to complete either of the two passes entirely. The reconstruction procedure,

© 2008 by Taylor & Francis Group, LLC



(a) Portion of Original (b) WDR

(c) J2K (d) ASWDR

FIGURE 4.13
Comparison of 32 to 1 compressions of *Barbara* image.

which just recapitulates the steps of the encoding based on the output data, can also terminate at any point. Therefore, progressive transmission and reconstruction are possible. For example, over the Internet, just enough data could be sent at first to produce a *thumbnail image* of high compression at the receiving end. Later this image could be updated to a higher resolution. In this way, webpages with a large number of images can be downloaded in progressive stages, allowing for faster access. Remotely scanning thumbnail images from a large database is another application of this progressive feature.

4.6.2 Lossless compression

For lossless compression, ASWDR, like J2K, uses the Daub 5/3 Integer-to-Integer transform. In ASWDR, this transform is additionally modified so as to be nearly energy preserving. This additional step is performed as follows.

© 2008 by Taylor & Francis Group, LLC

Suppose we perform a 1-level transform:

$$\mathbf{f} \longmapsto \begin{pmatrix} \mathbf{h}^1 & | & \mathbf{d}^1 \\ — & & — \\ \mathbf{a}^1 & | & \mathbf{v}^1 \end{pmatrix}.$$

We then modify the transform as follows:

$$\begin{pmatrix} \mathbf{h}^1 & | & \mathbf{d}^1 \\ — & & — \\ \mathbf{a}^1 & | & \mathbf{v}^1 \end{pmatrix} \longmapsto \begin{pmatrix} 2\mathbf{h}^1 & | & \mathbf{d}^1 \\ — & & — \\ 2\mathbf{a}^1 & | & 2\mathbf{v}^1 \end{pmatrix}$$

where, for instance, $2\mathbf{h}_1$ stands for the subimage obtained from $\mathbf{h}_1$ by multiplying all its values by 2. Notice that the $\mathbf{d}^1$ subimage is *not* multiplied by 2. For most images this modified transform is nearly energy-preserving. Typically, the modified transform has an energy that is between 100% to 103% of the energy of the original image. For higher level transforms, this procedure is iterated. For example, if we have a 4-level transform, then the subimages of the final modified transform are

$$16\mathbf{a}_4, 16\mathbf{h}_4, 16\mathbf{v}_4, 8\mathbf{d}_4, 8\mathbf{h}_3, 8\mathbf{v}_3, 4\mathbf{d}_3, 4\mathbf{h}_2, 4\mathbf{v}_2, 2\mathbf{d}_2, 2\mathbf{h}_1, 2\mathbf{v}_1, \mathbf{d}_1.$$

ASWDR typically performs a 4-level modified Daub 5/3 Integer-to-Integer transform of an image.[10] To see how well ASWDR performs lossless compression, we compare it to a standard lossless compressor of data, the WinZip® program. The results are shown in Table 4.6.

Table 4.6 Lossless compression

Image/Method	WinZip®	ASWDR
Barbara	10%	39%
Boats	23%	45%
X-ray	45%	68%

WinZip® uses the *Lempel-Ziv* method of data compression. That method is a general purpose lossless compressor for arbitrary data. It is not designed specifically for compressing images, so it is no wonder that ASWDR significantly outperforms it on each of the tested images. (Note: In the Exercises we compare ASWDR with a different lossless compression program, WinRAR®.)

There are some lossless image compressors that achieve slightly higher percentage compressions than both ASWDR and J2K. But those compressors do not have the progressive transmission/reconstruction feature that both ASWDR and J2K have. With this progressive transmission/reconstruction feature, a highly compressed thumbnail image (not quite as good quality as with Daub 9/7 but usually acceptable) can be reconstructed first, followed

[10] Just 4 levels are used since then the Daub 5/3 transform of an 8-bit image can be stored in a 16-bit array, thereby saving on memory and power utilization.

© 2008 by Taylor & Francis Group, LLC

by enough further data to achieve lossless reconstruction. In fact, both the lossy and lossless version of ASWDR (and J2K) even allow for higher quality reconstruction of just a portion of the thumbnail image selected by the viewer.

4.6.3 Region-of-interest

In many applications, such as medical imagery or reconnaissance, it would be of advantage to first view a highly compressed reconstruction (which can be transmitted rapidly) and then reconstruct at higher quality a select portion of the image. This portion is called a Region-of-Interest (ROI). In Figure 4.14 we provide an example. This example illustrates a reconnaissance image of an airfield taken from a spy plane. In Figure 4.14(a) we show the original image, which we assume is too large to efficiently transmit to an observer at another location. Hence, a 200:1 compression is performed (using a 4-level modified Daub 5/3 Integer-to-Integer transform). We show in Figure 4.14(b) the reconstruction of this compression at the receiving end. Suppose our observer now wishes precise details of the aircraft at lower left center, which after transmitting a request back to the spy plane, indicates the ROI outlined by the white rectangle in Figure 4.14(c). The spy plane then sends just enough data to reconstruct the ROI to exact, lossless detail, as shown in Figure 4.14(d). Some data should help us appreciate the efficiency gained from this approach. The original image size is 262,159 bytes, with 200:1 compression that means sending 1,282 bytes, which can be done quite rapidly. The update of just the ROI requires 11,427 bytes, as opposed to the 186,929 bytes that would be needed if the whole image was losslessly compressed. Clearly, when the communication channel is very low capacity, as it often is in reconnaissance, this approach is of real value. It also has applications in medical imagery, for instance, when huge images like mammograms need to be sent over low-capacity phone lines.

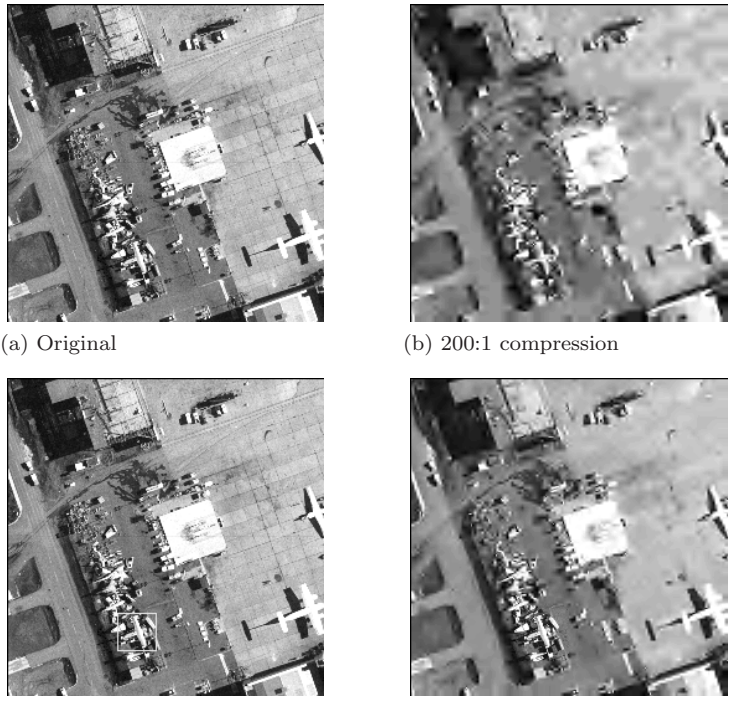
ASWDR (and J2K) have this ROI property because they encode precise positions of wavelet transform values. Consequently, when updating (in either the lossy or lossless case) *only transform values corresponding to wavelets whose supports overlap the ROI need to be considered.* When the ROI is relatively small, compared to the whole image, this provides a huge savings in data that needs to be transmitted.

4.7 JPEG 2000 image compression

The J2K image compression standard is an extremely complex algorithm. We cannot within the space of this primer give anywhere near a complete description. All we can do is provide an outline of the main aspects of the algorithm. For more complete details, consult the book by Taubman and Marcellin in the References.

J2K has all of the three features that we described for ASWDR at the end of the last section. In addition, because J2K does not work with the entire

© 2008 by Taylor & Francis Group, LLC



(a) Original (b) 200:1 compression

(c) Original with ROI marked (d) Update with Lossless ROI

FIGURE 4.14
Illustration of ROI capability.

wavelet transform held at one time in RAM, it is suitable for devices with very low memory and for huge images. That is one of the most important features of J2K, its low memory overhead.

We now outline the main aspects of the J2K algorithm. First, J2K employs a 5-level Daub 9/7 wavelet transform for lossy compression and a Daub 5/3 integer-to-integer transform for lossless compression.[11] Once the wavelet transform of the image is computed, then J2K divides the transform into *blocks,* subimages of the transform. Each block is coded separately and only one block is held in memory at any one time (that's how J2K achieves its low memory overhead and large image handling capability). Values from blocks are coded in a priority that corresponds to what level their values belong to. The block that is given first priority in encoding during each coding pass is one that contains at least the highest level trend values. Because this first block is sufficient to produce a rough, if somewhat blurred, version of the original image, J2K does not suffer from blocking artifacts like JPEG.

J2K performs a significance pass and a refinement pass on each separate block. But, because the blocks are significantly smaller than the full size transform, J2K can efficiently perform what is known as *fractional bit-plane*

[11] Although JPEG 2000 Part 2 allows for other wavelet transforms to be used as well.

© 2008 by Taylor & Francis Group, LLC

encoding. To illustrate the idea of fractional bit-plane encoding, consider the very simple case of there being just enough bits left to encode one of two values, say 28 or 20 when the threshold is 16. Assume first that the same number of bits would be needed in either case; then our intuition tells us to encode the larger value first. This choice is borne out by considering square error. Before encoding, the square error due to the two values 20 and 28 is $20^2 + 28^2 = 1184$. If 28 is encoded at a quantized value of 16, then the square error that remains is $20^2 + 12^2 = 544$. While if 20 is encoded, the square error that remains is $4^2 + 28^2 = 800$. Thus, if the same number of bits are needed for each encoding, it is better to encode 28 because there is a greater reduction of square error.

Of course, the number of bits to encode the two values may be different. J2K employs a sophisticated algorithm that measures the tradeoff in terms of bits used vs. reduction of square error in order to decide what quarter of new significant values to encode for each block so as to successively reduce square error as much as possible (the first quarter reducing square error the most, the second quarter the next most, etc.). This fractional bit-plane encoding achieves higher PSNRs than ordinary bit-plane encoding when, as frequently happens, there are only sufficient bits left for encoding a portion of the new significant values for the last threshold.

Finally, we briefly sketch the method of arithmetic coding that J2K employs. J2K encodes data as to the sign and significance of unencoded values at each threshold T_k using several contexts. A given context for an unencoded value is chosen based on the signs and quantized values of adjacent values. Such data provide good predictors for the sign and quantized value (0 or $\pm T_k$) of the unencoded value. By providing a good prediction, the amount of data that needs to be sent to record the true quantized value is significantly reduced through arithmetic coding.

4.7.1 Compressing color images

We conclude our discussion of J2K by briefly noting how it handles color images. The simplest encoding of color images, as Red-Green-Blue (RGB) intensities, does not produce the most efficient compression. One reason for that is that our visual system is much less sensitive to blue hues than the other two hues. Therefore, if a color image is in RGB format, J2K first maps the RGB values to three new values: *luminosity* I which is a gray-scale quantity, and two color values C_1 and C_2. (The formulas for this mapping are described in the book by Taubman and Marcellin.) The luminosity I, which is an average of the R, G, and B values, has much more effect on our visual perception of the color image than the two color components C_1 and C_2. After performing this mapping, I, C_1, and C_2 are compressed as separate images by the J2K method outlined above.[12] Much greater compression can be done on the C_1 and C_2 images than on the I image because of the much lower

[12]The WDR or ASWDR algorithms also allow for compression of color images.

© 2008 by Taylor & Francis Group, LLC

sensitivity that our visual system has for variations in color. More details on color compression can be found in the references for this chapter.

4.8 Denoising images

In this section we shall describe some fundamental wavelet based techniques for removing noise from images. Noise removal is an essential element of image processing. One reason for this is that many images are acquired under less than ideal conditions and consequently are contaminated by significant amounts of noise. This is the case, for example, with many medical images. Another reason is that several important image processing operations, such as contrast enhancement, histogram equalization, and edge enhancement, work much better if random noise is absent.

The basic concepts of wavelet denoising of images are similar to those described previously for 1D audio signals in Section 3.6. By choosing a threshold that is a sufficiently large multiple of the standard deviation of the random noise, it is possible to remove most of the noise by thresholding wavelet transform values. We shall illustrate thresholding using a high-performance wavelet-based image denoising algorithm, the "Tree-adapted Wavelet Shrinkage" (TAWS) algorithm. We conclude the section with an example of removing "clutter noise," noise that is analogous to the pop noise that we looked at in Section 3.6. We will show how clutter noise can be removed from scanning tunnelling microscope images.

4.8.1 The TAWS algorithm

The TAWS algorithm is a simple, but highly effective, wavelet-based image denoising algorithm. It is described in great detail in the easily obtainable survey article [32] listed in the references, so we shall only briefly describe it here. It is based on Principles A and B from Section 4.5. Those principles are used to distinguish image transform values from noise values at fairly low thresholds, thereby producing sharply focused denoisings.

The TAWS algorithm begins by setting a *base threshold* $T_B = \sigma\sqrt{2\log M}$ where σ is the standard deviation of the noise (we explain how σ is estimated below) and M is the larger of the row and column dimensions of the image. The base threshold T_B is the one proposed by David Donoho and his collaborators in a famous paper on wavelet denoising [33], which showed that, with probability 1, noisy transform values will lie below T_B in magnitude.[13] The problem with using T_B as the threshold is that it produces blurry denoisings. See Figure 4.15. To remedy this problem the TAWS algorithm uses the threshold $T_T = T_B/8$. For transform values that lie between T_B and T_T it uses Principle A to distinguish noise values from image values (that is a

[13] Because with Gaussian normal noise, the probability of a noise value's magnitude lying above T_B is much less than $1/M$. Since there are no more than M values along any row or column, it follows that the probability is 1 that no noise value's magnitude lies above T_B.

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image

(c) Base threshold denoising

(d) TAWS denoising

FIGURE 4.15
Two denoisings of Boats image.

transform value is retained if its parent value is significant, otherwise it is set to 0). Principle B is also used; isolated transform values (transform values whose siblings are not significant) whose magnitudes lie between T_B and T_T are discarded (set to 0). Finally, a *shrinkage* function is applied to the magnitude $|x|$ of each transform value x. This shrinkage function is

$$S(|x|) = \begin{cases} |x| - T_T & \text{if } |x| \geq T_T \\ 0 & \text{if } |x| < T_T. \end{cases}$$

This shrinkage operation typically reduces the MSE of the reconstructed image (the reason why is explained in the papers of Donoho and his collaborators cited in the references). In Figure 4.15(d) we show the TAWS denoising of the noisy boats image. It is superior to the base threshold denoising, producing a sharper more focused image that is clearly more closely approximate of the original image.

4.8.2 Comparison with Wiener denoising

As a test of the performance of TAWS, we compare it to a standard commercial denoising algorithm, the `wiener2` algorithm provided by MatLab®. The

© 2008 by Taylor & Francis Group, LLC

`wiener2` method is an implementation of the classic method of Wiener denoising, which is known to be the best *linear* denoising algorithm. That is, among all linear transforms, the one which maximizes PSNR is the Wiener algorithm. The TAWS algorithm, because of thresholding and shrinkage, is certainly not a linear method so it can actually outperform (get higher PSNRs) the Wiener method.

In Table 4.7 we give PSNRs for TAWS and `wiener2` on some test images. TAWS outperforms `wiener2`, producing higher PSNRs in all cases (except for one image, where the PSNRs were equal), with much higher PSNRs in several cases. In Figure 4.16 we show the `wiener2` and TAWS denoisings of the Barbara image with a noise standard deviation of $\sigma = 16$. The PSNRs for these two denoisings are both 28.0 dB. A close examination of the two denoisings—especially in relatively constantly illuminated regions, such as Barbara's forehead, left cheek, and right arm—shows that the Wiener denoising retains more noise artifacts. (See also Figure 4.28 on p. 162.)

Table 4.7 Two Denoising Methods

Noise st. dev. $\sigma = 8$			
Image	*Noisy*	*Wiener*	*TAWS*
Barbara	30.1	30.4	32.0
Boats	30.1	32.9	33.6
X-ray	30.1	38.2	39.9
Noise st. dev. $\sigma = 16$			
Image	*Noisy*	*Wiener*	*TAWS*
Barbara	24.1	28.0	28.0
Boats	24.1	29.8	29.9
X-ray	24.3	32.2	35.6
Noise st. dev. $\sigma = 32$			
Image	*Noisy*	*Wiener*	*TAWS*
Barbara	18.3	24.0	24.4
Boats	18.2	25.0	26.47
X-ray	19.0	26.1	31.5

This retention of noise artifacts is even more clearly revealed in the denoisings shown in Figure 4.17. Here, to be fair to the Wiener method, we point out that it retains more image details than the TAWS denoising. Some people, in fact, do prefer the `wiener2` denoising because the boat images (especially the masts and rigging) are more sharply focused and better retained in the `wiener2` denoising despite the retention of more noise artifacts.

The TAWS method was discussed here because it is one of the simplest high-performance wavelet denoisers. More details about TAWS, and more sophisticated denoisers, can be found in the Notes and References section.

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image $\sigma = 16$

(c) `wiener2` denoising

(d) TAWS denoising

FIGURE 4.16
Two denoisings of Barbara image.

4.8.3 Estimation of noise standard deviation *

When performing denoising, by TAWS or any other method, the noise standard deviation σ needs to be estimated. In this optional section, we briefly describe how this is done.

When the noise is Gaussian normal, the fraction of noise values that lie between $-a$ and a is given by

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-a}^{a} e^{-x^2/2\sigma^2}\, dx.$$

The *median* m then satisfies

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-m}^{m} e^{-x^2/2\sigma^2}\, dx = 0.5.$$

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image $\sigma = 32$

(c) `wiener2` denoising

(d) TAWS denoising

FIGURE 4.17
Comparison of two denoisings of Boats image.

If $\sigma = 1$, then $m = 0.6745$. By change of scale, for general σ, we obtain $m = 0.6745\sigma$. For the first-level fluctuation $\mathbf{d}_1$, most of the values are just noise. It follows that the standard deviation can be estimated by the median m_1 of the first-level fluctuation values divided by 0.6745. In experiments with images we have found that this estimator is typically within $\pm 1\%$ of the true standard deviation.

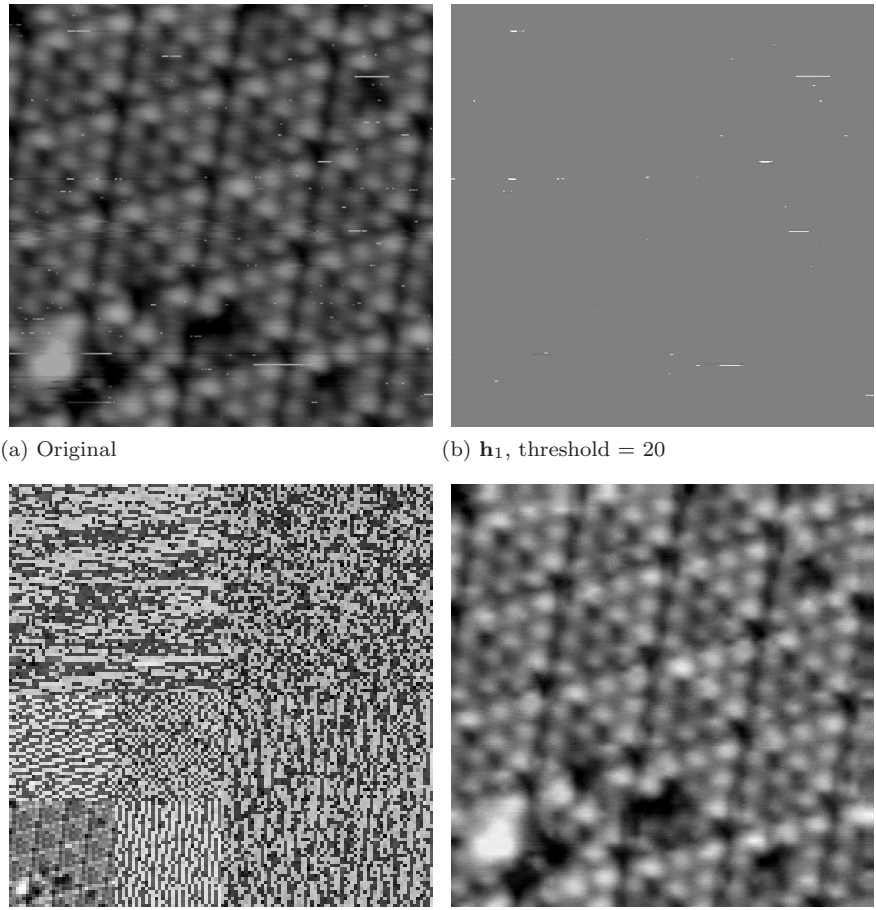
4.8.4 Removal of clutter noise

We close our treatment of denoising with an important scientific example, the removal of random clutter from Scanning Tunnelling Microscope (STM) images. The noise that occurs here is not random Gaussian, but rather appears as unwanted dots and horizontal bars, with maximum intensity 255, in the gray-scale STM images. See Figures 4.18(a) and 4.19(a). Those images are STM images of atoms of silicon arranged in crystal lattices.[14]

We now explain how we denoised these STM images. Unlike Gaussian noise, the clutter in the STM images produces extremely large fluctuation

[14]Thanks to Dr. Marcus T. McEllistrem, UWEC Chemistry Dept., for these images.

© 2008 by Taylor & Francis Group, LLC



(a) Original

(b) $\mathbf{h}_1$, threshold = 20

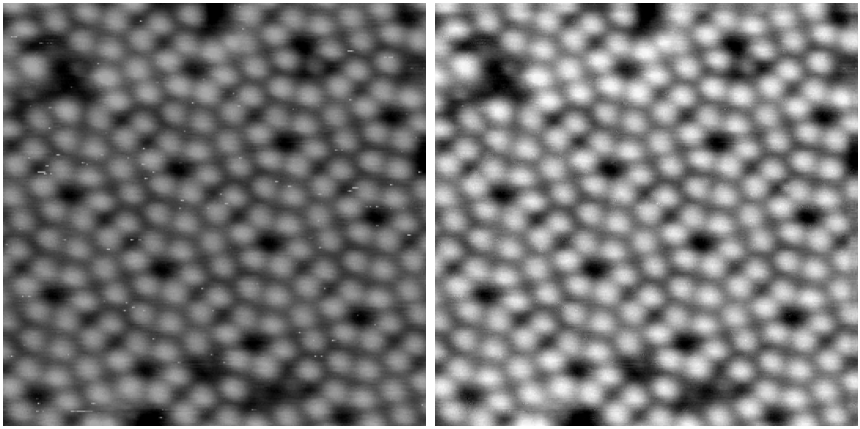
(c) Levels 5 and 4

(d) Denoised image

FIGURE 4.18
Removal of clutter noise from STM image.

values, typically much larger than the fluctuation values from the image. For instance, see Figure 4.18(b) where we show Daub 9/7 fluctuation values from $\mathbf{h}_1$ which are larger in magnitude than the threshold 20. Only fluctuation values that are due to the random clutter are above this threshold in magnitude. We also observe that the Daub 9/7 fifth level trend $\mathbf{a}_5$ is relatively free of noise, as one can see in the bottom left corner of Figure 4.18(c). Hence, we performed a 5-level Daub 9/7 transform of the noisy image, and did no modification of $\mathbf{a}_5$. We removed (set equal to 0) all fluctuation values at the first through fourth levels that had higher magnitude than 20. We found, however, that at the fifth level the fluctuation values traceable to the image were much larger than at other levels. Hence, at the fifth level we set all fluctuation values that were larger in magnitude than 350 equal to 0. After performing

© 2008 by Taylor & Francis Group, LLC



(a) Noisy image (b) Denoised image

FIGURE 4.19
Removal of clutter noise from STM image.

these modifications of the transform, we then performed an inverse transform, producing the denoised image shown in Figure 4.18(d). Because we were able to remove most of the noise, it is possible to magnify the brightness of the denoised image so as to better reveal the silicon atoms.

In Figure 4.19 we show an even more successful denoising. To produce the denoising in Figure 4.19(b), we modified a 5-level Daub 9/7 transform of the noisy image by simply setting to 0 all fluctuation values whose magnitudes were greater than the threshold 30. The denoised image could again be magnified in brightness, better revealing the lattice of silicon atoms.

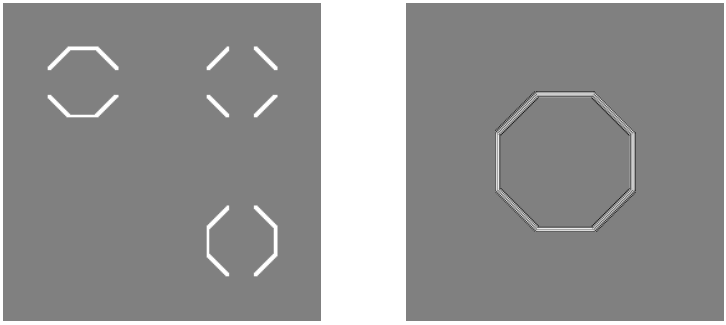
4.9 Some topics in image processing

We conclude our introduction to 2D wavelet analysis with a brief discussion of a few examples of wavelet based techniques in image processing. These examples are meant to provide the reader with a small sampling from a huge variety of applications. Further examples can be found in the references for this chapter.

4.9.1 Edge detection

One task in image processing is to produce outlines—the edges—of the various objects that compose an image. As shown in Figure 4.1, a wavelet transform can detect edges by producing significant values in fluctuation subimages. One way to produce an image of these edges is to simply set all the trend values equal to zero and then perform an inverse transform of only the fluctuation values (multiplied by a large factor). This produces an image of a multiple of the first-level detail image $\mathbf{D}^1$. For example, in Figure 4.20(a) we show such

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 4.20
(a) 1-level Coif6 transform of octagon image with trend subimage values all replaced by zeros and fluctuation values multiplied by 10. (b) Inverse transform of image in (a).

a modification of the 1-level Coif6 transform of the octagon image obtained by setting all its first trend values equal to zero and multiplying the first fluctuation values by 10. By performing an inverse transform of this modified transform, we produced the image $10\,\mathbf{D}^1$ shown in Figure 4.20(b). This figure clearly highlights the edges of the original image. Of course, this image could be further processed to increase the highlighting of just the outline of the octagon. For instance, only the most intense, whitest portion of the image could be retained, using a thresholding operation.

4.9.2 Edge enhancement

If we can detect edges accurately, then we can also enhance their appearance in an image. This will sharpen images that suffer from dull, blurry edges. For example, in Figure 4.21(a) we show an image of a house that suffers from blurred edges. In order to sharpen the edges of this image, we used the following method.

Edge Enhancement Method

Step 1. Perform a wavelet transform of the image.

Step 2. Multiply fluctuation values by a constant larger than 1, and leave trend values unchanged.

Step 3. Inverse transform the modified image from Step 2.

To produce the edge enhanced image of the house, we used a 1-level Daub4 transform in Step 1, and we multiplied the first-level fluctuation values by the constant 3 in Step 2.

Comparing the two images of the house in Figure 4.21, we can see that the edge enhanced image is a sharper image than the original. Some details

© 2008 by Taylor & Francis Group, LLC

(a) (b)

FIGURE 4.21
(a) Image of a house. (b) Edge enhanced image.

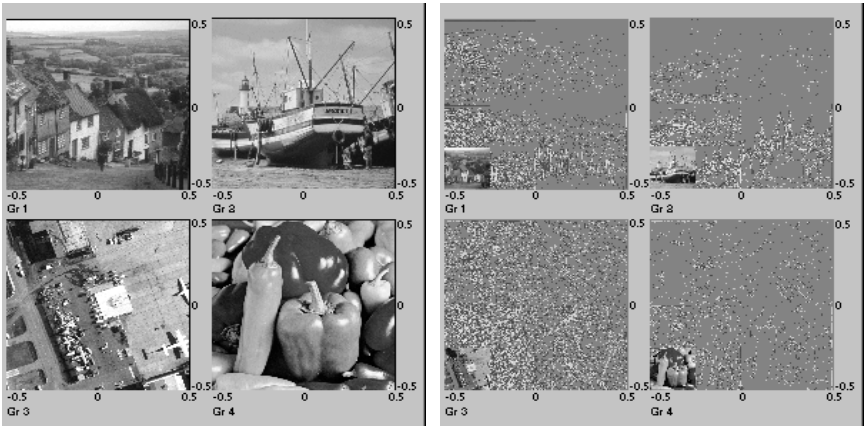
can be seen more clearly in the enhanced image, such as the fencing in front of the house, and the woodwork at the top of the roof. Particularly striking is the fact that in the right central window a vertical window divider has been rendered visible in the enhanced image. This kind of edge sharpening and improvement in detail visibility is clearly of fundamental importance in medical and biological imaging. The edge enhancement method just described is only the simplest of many related procedures. For instance, if a multiple level transform is performed in Step 1, then Step 2 can be done differently by using different constants for each fluctuation level.

4.9.3 Image recognition

Teaching a machine to recognize images is a very difficult problem. For example, we might want to program a computer to recognize images of human faces. Image recognition is important in areas such as data retrieval, object identification, and the science of human vision. *Data retrieval* refers to finding a match to a given image from a huge archive of images and then retrieving more data associated with that image (e.g., name, date of birth, etc.). *Object identification* refers to the problem of identifying the location, or lack thereof, of a particular object such as an aircraft within a complicated scene. Image retrieval relates to some aspects of the *science of human vision* because of our desire to know how our brains recognize images out of the data supplied by our eyes. While programming a computer to recognize an image is not the same as understanding how our brains do it, nevertheless, insights gained from the one task will certainly provide ideas for the other.

In Figure 4.22 we illustrate an elementary problem of image recognition. Suppose that we want a machine to identify which of the four images shown in Figure 4.22(a) is the best match to the denoised version of *Boats* in Figure 4.15(d). We choose to look for a match to the *denoised Boats,* rather than *Boats,* since it is more likely that we would not receive a precise image for matching, but rather a noisy version. It is also likely that the image we

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 4.22
(a) Four images. (b) 2-level Coif18 transforms of these images.

wish to match differs in other ways (different vantage point, different lighting conditions, etc.) from the image of that scene in the archive. This raises the considerably more difficult problem of matching distinct images of the same scene. It turns out that the wavelet based algorithm for image matching outlined below helps with this problem, too.

A simple method for matching would be to compute an error measure, say the *relative* 2*-norm error* $\mathcal{D}(\mathbf{f},\mathbf{g})$ defined by

$$\mathcal{D}(\mathbf{f},\mathbf{g}) = \frac{\sqrt{(f_{1,1}-g_{1,1})^2+(f_{1,2}-g_{1,2})^2+\cdots+(f_{N,M}-g_{N,M})^2}}{\sqrt{f_{1,1}^2+f_{1,2}^2+\cdots+f_{N,M}^2}}$$

where $\mathbf{f}$ is the denoised *Boats* image and $\mathbf{g}$ is each of the images in our archive. For the four images in Figure 4.22(a) this method works well. As shown in the column labeled *Full* in Table 4.8, the image labeled Gr 2 produces $\mathcal{D}(\mathbf{f},\mathbf{g}) = 0.056$, and all other images produce $\mathcal{D}(\mathbf{f},\mathbf{g}) > 0.4$. So the image in Gr 2 is clearly the best match. There is a significant problem with this approach, however, since an actual archive will certainly contain far more than four images—it might contain tens of thousands of images. The images we are using are 512 by 512; hence each image contains over a quarter of a million values. The time delay—due to the enormous number of calculations—associated with finding $\mathcal{D}(\mathbf{f},\mathbf{g})$ for tens of thousands of images is prohibitively large.

Wavelet analysis provides a flexible approach for significantly reducing the number of computations needed for this image matching problem. The wavelet based approach also incorporates its own compression procedure, which allows for the archive to be stored in a compressed format. To illustrate this wavelet approach to image matching, consider the 2-level Coif18 transforms of the four

© 2008 by Taylor & Francis Group, LLC

images in our small archive shown in Figure 4.22(b). Notice that our visual systems can easily distinguish the four images based only on their second trend subimages, which are one-sixteenth the size of the original image. If we simply compute the error measures for these second trends only, then we obtain the results shown in the column labeled *Second* in Table 4.8. This new computation easily matches the *denoised Boats* image with the correct image of *Boats* in our tiny archive. Since the second trends are one-sixteenth the size of the original images, this new computation is sixteen times faster than the first one involving the full images.

Table 4.8 Error measures for images

Image	*Full*	*Second*	*Third*	*Fourth*	*Fifth*
Gr 1	0.464	0.455	0.444	0.426	0.402
Gr 2	0.056	0.026	0.015	0.008	0.004
Gr 3	0.558	0.542	0.523	0.498	0.464
Gr 4	0.547	0.540	0.532	0.515	0.480

What works well for second-level trends also works well for third, fourth, and even fifth trends. As shown in Table 4.8, computing errors for any of these trends clearly determines the correct match. If we use fifth trends to perform error computations, then our calculations proceed over 1000 times faster than if we were to use the full images. This improved performance factor makes it quite practical to search through an archive of tens of thousands of images. It is important to note that in order to achieve this rapid performance, the images must be stored in the archive in their wavelet transformed format. Fortunately, this also allows for the images to be stored in a compressed format.

Another advantage of comparing these very low resolution, fifth trend images is the following. If the given image is only a fair approximation, rather than a perfect replica, of its version in the archive, then the low resolution fifth trend versions should still produce smaller error measures than the fifth trends of very different looking people. This indicates how a wavelet based algorithm facilitates the matching of distinct images of the same scene.

Of course, if we are using fifth trends it is quite possible that several images of different scenes might produce errors that are less than whatever error threshold we have established for matching. In that case, by using the fourth-level fluctuations, we can rapidly construct the fourth-level trend subimages for the subset of images which survived the fifth-level matching test. The error measures can then be computed for these fourth-level trends. Although these additional calculations require more time, it is very likely that the subset of images that survived the fifth trend comparison is much less numerous than the original set. The recursive structure of wavelet transforms enables the implementation of a recursive matching algorithm that successively winnows the subset of possible matching images, using extremely small numbers

© 2008 by Taylor & Francis Group, LLC

of computations to distinguish grossly dissimilar images, while reserving the most numerous computations only for a very small number of quite similar images.

This brief outline of an image retrieval algorithm should indicate some of the fundamental advantages that a wavelet based approach provides. One of the most important features of the approach we described is its use of the multiresolution structure (the recursive structure of the trend subimages) to facilitate computations. This has applications to many other problems. For instance, in object identification, a method known as *correlation,* which we will describe in the next chapter, is a computationally intensive method for locating a given object within a complicated scene. Using correlation on third trends instead of the full images, however, can greatly speed up the approximate location of an object.

4.10 Notes and references

Two good references on digital image processing are [1] and [2]. The book [3] contains several excellent papers on wavelets and their applications in biology and medicine. A summary of wavelet-based image compression prior to J2K can be found in [4]. Further details on wavelet-based image compression prior to J2K can be found in [5]. An archive of images can be downloaded from [6].

An excellent treatment of image compression in general, and J2K in particular, can be found in [7]. A briefer synopsis of J2K can be found in the webpage article [8]. Information on WDR from its creators is in [9] and [10]. The original papers on ASWDR are [11] and [12]. More discussion on the relation of ASWDR to visual aspects of compression is given in [11].

FAWAV performs the ASWDR algorithm. WDR can be performed using the software `ImageCompress` from the FAWAV installation. The J2K compressions were done using the software `Image Analyzer` which can be downloaded from [13]. WSQ compressions can be performed with the free software `WSQ Viewer` which can also be downloaded from [13]. Two excellent articles on WSQ, written by its developers, are [14] and [15].

There is an interesting analysis of parent/child structures in wavelet transforms and their relationship with wavelet-based and fractal-based image compression in [16]. The differences in sensitivity of the human visual system to luminosity versus color hues are described in [17]. The book [17] is the best elementary introduction to the science of the human visual system. In [17] there is a good summary of MRA and its relation to vision. Wavelet-like models of the human visual system are described in [18] to [21]. Important work on wavelet methods and their application to imaging and vision has been done by Eero Simoncelli and his collaborators. See, for example, [22] to [26].

Recent work on objective measures for the quality of image reconstructions (as in decompressions and denoisings) can be found in [27], [28], and [23].

A brief, but very lucid and informative, treatment of arithmetic compression is given in [29]. A thorough, and very readable, treatment of the subject

© 2008 by Taylor & Francis Group, LLC

can be found in the book by Bell, Cleary, and Witten [30]. Another excellent treatment, available for free downloading, can be found in [31]. It contains a nice proof that arithmetic coding achieves asymptotically the minimum average coding length (entropy).

The TAWS algorithm, and wavelet-based denoising in general, is described in detail in the survey article "Tree-adapted wavelet shrinkage" [32]. The papers of Donoho and his collaborators referred to in the section on denoising are [33] to [35]. State of the art denoising methods, using advanced statistical modelling of wavelet transforms, can be found in [36] to [38].

There is an interesting discussion in [39] of a wavelet based solution of the *rogues gallery problem,* the problem of retrieving an image from a large archive. This discussion differs somewhat from the one given here.

1. B. Jähne. (2002). *Digital Image Processing, 5th Ed.* Springer, New York, NY.
2. J.C. Russ. (1995). *The Image Processing Handbook.* CRC Press, Boca Raton, FL.
3. A. Aldroubi and M. Unser, editors. (1996). *Wavelets in Medicine and Biology.* CRC Press, Boca Raton, FL.
4. J.S. Walker and T.Q. Nguyen. (2000). Wavelet-based image compression. *Handbook of Transforms and Data Compression,* Chap. 6, 267–312, CRC Press, Boca Raton, FL.
5. G.M. Davis and A. Nosratinia. (1998). Wavelet-based image coding: an overview. *Applied and Computational Control, Signals and Circuits,* Vol. 1, 205–269.
6. Image archive. Available at

   http://www.uwec.edu/walkerjs/Primer/Data/Primer_Images.zip

7. D.S. Taubman and M.W. Marcellin. (2002). *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Kluwer, Boston, MA.
8. J. Li. (2002). Image compression—The mechanics of the JPEG 2000. Available at

   research.microsoft.com/~jinl/paper_2002/msri_jpeg.htm

   or at the primer website [13].
9. J. Tian and R.O. Wells, Jr. (1996). A lossy image codec based on index coding. *IEEE Data Compression Conference, DCC '96,* p. 456.
10. J. Tian and R.O. Wells, Jr. (1998). Embedded image coding using wavelet difference reduction. *Wavelet Image and Video Compression,* P. Topiwala, Ed., Kluwer, Norwell, MA, 289–301.
11. J.S. Walker. (2000). Lossy image codec based on adaptively scanned wavelet difference reduction. *Optical Engineering,* Vol. 39, 1891–1897.
12. J.S. Walker and T.Q. Nguyen. (2000). Adaptive scanning methods for wavelet difference reduction in lossy image compression. *IEEE Int'l Conference on Image Processing, Vancouver, Sept. 2000,* Vol. 3, 182–185.
13. Book website: http://www.uwec.edu/walkerjs/Primer/

© 2008 by Taylor & Francis Group, LLC

14. J.N. Bradley, C.M. Brislawn, and T. Hopper. (1993). The FBI Wavelet/Scalar Quantization Standard for gray-scale fingerprint image compression. *SPIE,* Vol. 1961, *Visual Information Processing II (1993)*, 293–304.

15. C.M. Brislawn. (1995). Fingerprints go digital. *Notices of the Amer. Math. Soc.,* Vol. 42, 1278–1283.

16. G.M. Davis. (1998). A wavelet-based analysis of fractal image compression. *IEEE Trans. on Image Proc.,* Vol. 7, 141–154.

17. B.A. Wandell. (1995). *Foundations of Vision.* Sinauer Associates, Sunderland, MA.

18. A.B. Watson. (1987). Efficiency of a model human image code. *J. Optical Soc. Am.,* Vol. 4, 2401–2417.

19. D. Field. (1993). Scale invariance and self-similar "wavelet" transforms: an analysis of natural scenes and mammalian visual systems. In *Wavelets, Fractals and Fourier Transforms,* M. Farge, J. Hunt, J. Vassilicos, Eds. Clarendon Press, Oxford, UK, 151–193.

20. D. Field. (1994). What is the goal of sensory coding? *Neural Computations,* Vol. 6, 559–601.

21. D. Field. (1999). Wavelets, vision and the statistics of natural scenes. *Phil. Trans. R. Soc. London A,* Vol. 357, 2527–2542.

22. S. Lyu and E.P. Simoncelli. (2007). Statistical modeling of images with fields of Gaussian scale mixtures. In *Adv. Neural Information Processing Systems,* Vol. 19.

23. Z. Wang, A.C. Bovik, and E.P. Simoncelli. (2005). Structural approaches to image quality assessment. Chapter 8.3 in *Handbook of Image and Video Processing, 2nd Edition,* Alan Bovik (Ed.), Academic Press, San Diego, CA.

24. Z. Wang and E.P. Simoncelli. (2004). Local phase coherence and the perception of blur. *Adv. Neural Information Processing Systems,* Vol. 16.

25. E.P. Simoncelli and B.A. Olshausen. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience,* Vol. 24, 1193–1216.

26. D.J. Heeger, E.P. Simoncelli, and J.A. Movshon. (1996). Computational models of cortical visual processing. *Proc. National Academy of Science,* Vol. 93, 623–627.

27. Z. Wang and A.C. Bovik. (2002). A universal image quality index. *IEEE Signal Processing Letters,* Vol. 9, 81–84.

28. Image quality indices webpage: Available at `http://www.cns.nyu.edu/~zwang/`

29. M. Nelson. (1991). Arithmetic Coding + Statistical Modeling = Data Compression. Available at

    `http://www.dogma.net/markn/articles/arith/part1.htm`

30. T.C. Bell, J.G. Cleary, and I.H. Witten. (1990). *Text Compression.* Prentice Hall, Englewood Cliffs, NJ.

© 2008 by Taylor & Francis Group, LLC

31. A. Said. (2004). *Introduction to Arithmetic Coding – Theory and Practice.* Chap. 5 of *Lossless Compression Handbook* (2003), K. Sayood (Ed.), Academic Press, San Diego, 101–152. Available at

http://www.hpl.hp.com/techreports/2004/HPL-2004-76.pdf

32. J.S. Walker. (2002). Tree-adapted wavelet shrinkage. *Advances in Imaging and Electron Physics,* Vol. 104, 343–394.

33. D. Donoho and I. Johnstone. (1994). Ideal spatial adaptation via wavelet shrinkage. *Biometrika,* Vol. 81, 425–455.

34. D. Donoho and I. Johnstone. (1995). Adapting to unknown smoothness via wavelet shrinkage. *American Statistical Assoc.,* Vol. 90, 1200–1224.

35. D. Donoho et al. (1995). Wavelet shrinkage: asymptopia? *J. Royal Stat. Soc. B,* Vol. 57, 301–369.

36. J. Portilla, V. Strela, M. Wainwright, and E.P. Simoncelli. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Processing,* Vol. 12, 1338–1351.

37. D. Hammond and E. Simoncelli. (2005). Image denoising with an orientation-adaptive Gaussian scale mixture model. To appear in *Proc. 13th IEEE Int'l Conference on Image Processing,* Atlanta, Georgia.

38. F. Luisier, T. Blu, and M. Unser. (2007). A new SURE approach to image denoising: inter-scale orthonormal wavelet thresholding. *IEEE Trans. Image Processing,* Vol. 16, 593–606.

39. M.V. Wickerhauser. (1994). *Adapted Wavelet Analysis from Theory to Software.* A.K. Peters, Wellesley, MA.

4.11 Examples and exercises

Section 4.1

Example 4.1.A For the array

$$\begin{pmatrix} 4 & 8 & 8 & 4 \\ 4 & 8 & 6 & 6 \\ 4 & 0 & 0 & 4 \\ 8 & 8 & 4 & 4 \end{pmatrix}$$

we compute its 1-level Haar transform in two steps. First, we find the 1-level Haar transform along each row:

$$\begin{pmatrix} 6\sqrt{2} & 6\sqrt{2} & -2\sqrt{2} & 2\sqrt{2} \\ 6\sqrt{2} & 6\sqrt{2} & -2\sqrt{2} & 0 \\ 2\sqrt{2} & 2\sqrt{2} & 2\sqrt{2} & -2\sqrt{2} \\ 8\sqrt{2} & 4\sqrt{2} & 0 & 0 \end{pmatrix}.$$

© 2008 by Taylor & Francis Group, LLC

Second, we find the 1-level Haar transform of each column of this new array (indexing from the bottom up):

$$\begin{pmatrix} 0 & 0 & 0 & -2 \\ 6 & 2 & -2 & 2 \\ 12 & 12 & -4 & 2 \\ 10 & 6 & 2 & -2 \end{pmatrix}$$

which is the 1-level Haar transform.

Example 4.1.B [Figure 4.1] To graph Figure 4.1(a), you select *New 2 dim* from the menu and plot the function

```
(y-x-c<=0)(x+y-c<=0)(x-y-c<=0)
(x+y+c>=0)(abs(y)<=b)(abs(x)<=b)
 \c=12/5\b=8.5/5\Rem Use L = 4
```

over $[-L, L] \times [-L, L]$ using $L = 4$. You then right-click on the graph and select *Graph style* which allows you to replot the graph using the *Grey (+/-)* option. To get the 1-level Coif6 transform in Figure 4.1(b), you plot a Coif6 transform of the octagon graph in (a), and then change the *Graph style* of the transform to *Grey (+/-)* with the `LinLog` option selected and with a threshold of .0000000001. The graphs in (c) and (d) are obtained in a similar way, except that 2-level and 3-level Coif6 transforms are performed, respectively.

Example 4.1.C [Figure 4.2] To graph Figure 4.2(a), you select *New 2 dim* from the *File* menu and then select *Points/128* from the *Edit* menu. You then graph the formula

```
del(x+64-2)del(y+32-4)
```

over $[-L, L] \times [-L, L]$ using $L = 64$. That produces an image with all values 0, except one pixel of value 1 (an element of the standard basis for 128 by 128 matrices), which when an inverse Haar transform is applied produces a Haar wavelet. Then, you perform a 2-level *inverse* Haar transform, and right-click on the inverse transform's graph in order to select *Graph style* which allows you to replot the transform using the *Grey (+/-)* option. That completes the construction of the image in (a). The images in Figures (b) to (d) were obtained in a similar way through modifying the formula above (use the formulas in the archive `Primer_Figures.zip`).

4.1.1 For each of the following arrays, compute its 1-level Haar transform.

(a)$_{\mathbf{s}}$ $\begin{pmatrix} 2 & 4 & 2 & 0 \\ -2 & 0 & 2 & 2 \\ 4 & 0 & 2 & 4 \\ 6 & 8 & 12 & 12 \end{pmatrix}$

(b) $\begin{pmatrix} 1 & 3 & 3 & 1 \\ 2 & 5 & 5 & 2 \\ 4 & 7 & 7 & 4 \\ 2 & 7 & 7 & 2 \end{pmatrix}$

(c) $\begin{pmatrix} 3 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 3 & 5 & 7 & 9 \\ 9 & 7 & 5 & 3 \end{pmatrix}$

© 2008 by Taylor & Francis Group, LLC

(d) $\begin{pmatrix} 4 & 6 & 8 & 12 \\ 4 & 16 & 24 & 32 \\ 8 & 12 & 6 & 6 \\ 4 & 4 & 2 & 2 \end{pmatrix}$

4.1.2$_s$ For each of the arrays in problem 4.1.1, compute 2-level Haar transforms.

4.1.3$_s$ Show that $h_{1,1} = \mathbf{f} \cdot (\mathbf{V}_1^1 \otimes \mathbf{W}_1^1)$.

4.1.4 Compute $\mathbf{W}_1^1 \otimes \mathbf{W}_1^1$, where $\mathbf{W}_1^1$ is a Haar wavelet.

4.1.5$_s$ Compute $\mathbf{W}_1^1 \otimes \mathbf{V}_1^1$, where $\mathbf{W}_1^1$ and $\mathbf{V}_1^1$ are a Haar wavelet and Haar scaling signal, respectively. Show that $v_{1,1} = \mathbf{f} \cdot (\mathbf{W}_1^1 \otimes \mathbf{V}_1^1)$.

4.1.6^c Compute $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$, $\mathbf{V}_3^2 \otimes \mathbf{W}_5^2$, $\mathbf{W}_3^2 \otimes \mathbf{V}_5^2$, and $\mathbf{W}_3^2 \otimes \mathbf{W}_5^2$, when $\mathbf{V}_j^2$ and $\mathbf{W}_k^2$ are Daub 9/7 scaling signals and Daub 9/7 wavelets, respectively.

4.1.7 Suppose a 1-level wavelet transform of an image $\mathbf{f}$ has $\mathbf{h}^1 = 0$, $\mathbf{a}^1 = 0$, and $\mathbf{v}^1 = 0$ (i.e., all the values in these sub-arrays are zero), and suppose that $d_{2,3}^1 = 1$ while all other values of $\mathbf{d}^1$ are zero. Show that the inverse transform yields $\mathbf{f} = \mathbf{W}_2^1 \otimes \mathbf{W}_3^1$. How does this result generalize?

Note: For subsequent examples and exercises, you may need to download images that are not installed by your initial installation of FAWAV. These additional image files can be downloaded from the *Audio, Images, Formulas* link at the book's website.

Section 4.2

Example 4.2.A [Figure 4.3] To generate the 4:1 compression in the figure you do the following. First, right-click on an image square on the 2D-form and select *Load/Image* and load the PGM file `barb.pgm`. You then select *File/Save options (images)* and enter a *target rate* of 2 bpp. Since the original image is 8 bpp that represents a 4:1 compression ratio. After saving a compressed image (right-click on the original image, choose *Save/Image*, and select `*.wc2` as file format), you then load this compressed file (*Load/Image* with `*.wc2` as format) and that produces the reconstruction shown in part (b) of Figure 4.3. The image in (c) was produced by plotting the function `g1 - g2` where `g1` stands for graph 1 (the original image) and `g2` stands for graph 2 (the reconstruction of the 4:1 compression). This third image is displayed with a *Graph style* of *Grey (+/-)* selected. Finally, the fourth graph in (d) was obtained by right-clicking on the graph in (c), and then plotting a 9-bit histogram with *Include sign bit* checked.

Example 4.2.B [Figure 4.4] To generate the 16:1 compression in the figure you load the image `dog_head.pgm` then select *File/Save options (images)* and enter a *target rate* of 0.5 bpp. (Since the original image is 8 bpp that represents a 16:1 compression ratio.) After saving a compressed image (right-click on the original image, choose *Save/Image*, and select `*.wc2` as file format), you then load this compressed file (*Load/Image* with `*.wc2` as format) and that produces the reconstruction shown in part (b) of Figure 4.4.

© 2008 by Taylor & Francis Group, LLC

Example 4.2.C [**Figure 4.5**] The image in (a) was obtained by loading the file `boat.bmp`. To obtain the reconstruction of the 64:1 JPEG compression shown in (b) we used the program IMAGE ANALYZER.[15] We opened the image file `boat.bmp` with IMAGE ANALYZER and then selected *File format options* from the *File* menu. Clicking on the *JPEG* tab and entering 4 kb for the compressed file size, we then saved the image as a `*.jpg` file. Loading that compressed `*.jpg` file produced the reconstruction shown in (b). The image in (c) was produced in a similar way, except that the *JPEG 2000* tab was clicked and we entered 1.5625 for the *% of compressed size*, and the *extra compression option* of `mode=real` was entered (that specifies that a Daub 9/7 transform is to be used). The image was then saved and opened as a `*.jp2` file. To obtain the image in (d) we used FAWAV. After loading the `boat.bmp` image, we selected *Save options (images)* from the file menu and entered 0.125 for the bpp rate. We then saved the image (right-clicking on the image and selecting *Load/Save*) as a `*.wc2` file. Loading that saved `*.wc2` file produced the image shown in (d).

Note: The PSNRs in the text were obtained by selecting *Analysis/Norm Difference* and then choosing the option *PSNR*. For example, to obtain the PSNR between image 2 (Gr 2) and image 3 (Gr 3), where image 2 is the original being compared with, you would enter **2** for the *Graph 1* number and **3** for the *Graph 2* number, select option *PSNR*, and click on the *Compute* button. Unfortunately, FAWAV does not support the `*.jp2` file format. Therefore, to obtain PSNRs for `*.jp2` reconstructions, you must first save a reconstructed (decompressed) `*.jp2` image in a file format that FAWAV can read (such as `*.bmp`).

4.2.1$^{c}_{s}$ Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `Airfield.bmp` image.

4.2.2^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `goldhill.bmp` image.

4.2.3^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `peppers.bmp` image.

4.2.4^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `zelda.bmp` image.

Section 4.3

Example 4.3.A [**Figure 4.6**] To obtain the image in part (a) of Figure 4.6, we loaded the image `fingerprint 1.bmp` and then right-clicked on a point with coordinates $(227, 158)$ within the image and selected *Zoom*. Images (b) to (d) were obtained by reconstructing 20:1 compressions by the JPEG, J2K, and ASWDR

[15] IMAGE ANALYZER can be obtained as a free download from the *Software* link at the book's website.

© 2008 by Taylor & Francis Group, LLC

methods, respectively and then zooming around the same coordinates as for image (a).

Example 4.3.B [Figure 4.7] To obtain the image in part (a) of Figure 4.7, we loaded the image `fingerprint 1.bmp`. The image in (b) was obtained from reconstructing a 0.8 bpp ASWDR compression of the image in (a). To obtain the images in (c) and (d) we zoomed around the coordinates $(157, 233)$ for images (a) and (b), respectively. *Note:* If PSNR is calculated for these zoomed images (by selecting *Analysis/Norm difference* and entering the graph numbers for the zoomed images), we find that it is 31.1 dB (which is lower than the PSNR for the full images, but still above the rule-of-thumb value of 30 dB).

4.3.1^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find PSNRs for 8:1, 16:1, and 32:1 compressions of the `fingerprint 1.bmp` image.

4.3.2$^{c}_{s}$ In problem 4.3.1, find PSNRs for zoomed images around the coordinates $(157, 233)$.

4.3.3^{c} Repeat problem 4.3.1, but use the image `fingerprint_2.bmp`.

4.3.4^{c} Using the image `fingerprint_2.bmp`, repeat problem 4.3.1, but for zoomed images around the coordinates $(137, 96)$.

Section 4.4

Example 4.4.A [Figure 4.10] To produce Figure 4.10(a) you load the image `boat.pgm`, select *Transform/Wavelet,* and then plot a 5-level Daub 9/7 wavelet transform of the image. You then select *Graph/Plot* and plot the formula `abs(g2)>=64` where `g2` stands for the wavelet transform (graph number `2`). That produces the image shown in (a), the significance map for threshold 64. To produce the image in (d), the approximate image reconstructed for threshold 64, you plot the graph `g2(abs(g2)>=64)` and compute the 5-level inverse Daub 9/7 wavelet transform of the image produced by that plot. The images in (b), (e), (c), and (f) are produced by obvious modifications of the methods just described.

Example 4.4.B [Table 4.5] To produce the PSNRs in the *WDR (No AC)* column for the BARBARA IMAGE portion of the table, we used the program IMAGECOMPRESSOR located in the main FAWAV directory. Using the choice *Get Image* on the *File* menu of IMAGECOMPRESSOR we loaded the image `Barb.pgm` and then performed compressions and decompressions in the following way. After selecting *Compress,* to perform *WDR (No AC)* we selected the options *Binary* and *WDR.* We also selected a Daub 9/7 transform with 5 levels and entered 1 for the *Bit rate (bpp)* value. By clicking the *Compress* button, we were then able to save to an 8:1 compressed file. By selecting *Decompress* from the file menu, we then decompressed this file and then selected *Error measures* to compute a PSNR between the original and reconstructed image. To get the 16:1 and 32:1 PSNRs, we *did not do further compressions.* Instead, we decompressed the 1 bpp compressed file at 0.5 bpp (for 16:1) and 0.25 bpp (for 32:1). To find the PSNRs for *WDR* we proceeded as just described, except that we checked the option *Arithmetic* when performing the 1 bpp compression. For *ASWDR (No AC),* you select *ASWDR* and *Binary.* For *ASWDR,*

© 2008 by Taylor & Francis Group, LLC

you select *ASWDR* and *Arithmetic.* The PSNRs for the BOATS IMAGE and X-RAY IMAGE parts of the table were computed by repeating all this work for the images `Boat.pgm` and `dog_head.pgm` respectively.

Example 4.4.C [Huffman compression] A simpler method of compression than arithmetic compression is the method called *Huffman compression.* The best way to describe Huffman compression is by an example. Suppose that we have 5 symbols to encode, say $\{a, b, c, d, e\}$, with the frequencies $9/30, 8/30, 6/30, 4/30, 3/30$, respectively, as shown in the left-most column in Figure 4.23. The method of creating the Huffman encoding is to merge the two least probable symbols; in the first instance that would be d and e, to create a combined symbol $\{d, e\}$ with probability $4/30 + 3/30 = 7/30$. The bit 1 is assigned to the branch connecting the higher frequency symbol d to the merged symbol $\{d, e\}$, and the bit 0 is assigned to the branch connecting the lower frequency symbol e to the merged symbol $\{d, e\}$ (as shown in Figure 4.23). [If the two symbols have the same frequency then the 1 bit is assigned to the first listed symbol in the tree diagram.]

This merging process continues, as illustrated in the figure, until the end when all symbols have been merged into the set $\{a, b, c, d, e\}$ with probability 1. Tracing back from this end node, we can read off the bits on the connecting branches to the symbols, thereby obtaining the following Huffman encoding:

Symbol	*Encoding*
a	1 1
b	1 0
c	0 0
d	0 1 1
e	0 1 0

(The reader might find it interesting to express the bits in these encodings as answers to yes/no questions.) The average length of an encoding with this Huffman code is

$$\frac{9}{30} \cdot 2 + \frac{8}{30} \cdot 2 + \frac{6}{30} \cdot 2 + \frac{4}{30} \cdot 3 + \frac{3}{30} \cdot 3 = 2.23333\ldots$$

which is very close to the theoretical minimum (entropy) value of

$$\frac{9}{30} \cdot \log_2\left(\frac{30}{9}\right) + \frac{8}{30} \cdot \log_2\left(\frac{30}{8}\right) + \frac{6}{30} \cdot \log_2\left(\frac{30}{6}\right) + \frac{4}{30} \cdot \log_2\left(\frac{30}{4}\right) + \frac{3}{30} \cdot \log_2\left(\frac{30}{3}\right) = 2.21375\ldots$$

This example illustrates that Huffman encoding of these symbols, with the given frequencies, would be just about as good as arithmetic coding. It is known that Huffman encoding produces the shortest average length encoding when a fixed set of integer-length codes are used for encoding. In many cases, arithmetic coding with its ability to assign variable (fractional average) length codes to symbols can outperform the Huffman approach. For example, if we have two symbols a and b with frequencies 0.9 and 0.1 then we have shown that the sequence $a\,a\,a\,a\,a\,a\,b$ could be arithmetically encoded with just 4 bits. The Huffman code for this case would be 1 for a and 0 for b, and that would use 7 bits for this same sequence. The arithmetic

© 2008 by Taylor & Francis Group, LLC

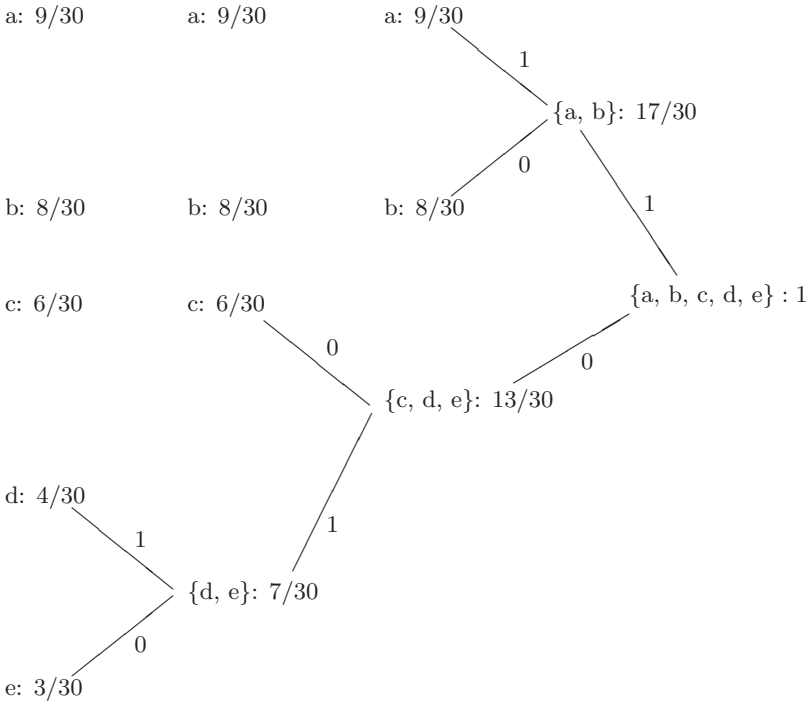


FIGURE 4.23
Tree Diagram for Huffman Compression.

code is almost twice as short for this example. In fact, the average coding length for an arithmetic code in this case is

$$0.9 \cdot \log_2(1/0.9) + 0.1 \cdot \log_2(1/0.1) = 0.46899559\ldots$$

while the average length for Huffman coding is

$$0.9 \cdot 1 + 0.1 \cdot 1 = 1$$

so arithmetic coding is slightly more than twice as efficient as Huffman encoding. Moreover, as we discuss briefly in Section 4.5, arithmetic coding is adaptable to much more sophisticated statistical modeling of symbol sequences (beyond just frequencies of single symbols viewed in isolation) and that allows for even higher compression—this fact, along with the simple example we just gave, illustrates why arithmetic compression is now the standard technique in image compression (and data compression in general).

4.4.1$_s$ Find the quantized transform for the wavelet transform in Figure 4.8(b), when the threshold is 2. Also find the analog of the last stage, half-threshold, array shown in Figure 4.9(b) when the threshold is 2.

© 2008 by Taylor & Francis Group, LLC

−4	−4	−12	−12	3	−4	−2	6
10	10	10	12	1	2	2	−4
−6	−6	−6	−8	4	−2	5	−8
4	4	6	4	2	4	−1	−2
4	4	5	−3	4	5	4	4
6	6	−3	5	−9	−8	−8	−7
16	20	−10	8	−9	−9	−10	−10
18	18	−10	8	8	6	6	8

FIGURE 4.24
2-level wavelet transform for Exercise 4.4.4.

32	31	30	29	55	61	62	64
25	26	27	28	54	56	60	63
24	23	22	21	50	53	57	59
17	18	19	20	49	51	52	58
8	7	14	16	36	37	44	45
5	6	13	15	35	38	43	46
2	4	9	12	34	39	42	47
1	3	10	11	33	40	41	48

(a) Scan order

−4	−4	−12	−12	3	−4	−2	6
10	10	10	12	1	2	2	−4
−6	−6	−6	−8	4	−2	5	−8
4	4	6	4	2	4	−1	−2
4	4	5	−3	4	5	4	4
6	6	−3	5	−9	−8	−8	−7
16	20	−10	8	−9	−9	−10	−10
18	18	−10	8	8	6	6	8

(b) Wavelet transform

FIGURE 4.25
Data for Exercises 4.4.4 and 4.4.5. (a) 2-level scan order. (b) 2-level wavelet transform.

4.4.2 Find the quantized transform for the wavelet transform in Figure 4.8(b), when the threshold is 1. Why is the half-threshold, last stage approximation unnecessary when the threshold is 1?

4.4.3 For the 2-level wavelet transform shown in Figure 4.24, find the quantized wavelet transforms for thresholds 16, 8, and 4.

4.4.4$_s$ Using the scan order in Figure 4.25(a), compute the wavelet difference reduction encoding (using symbols $+, -, 0, 1$) for the first pass (threshold 16) of the wavelet transform on the right of Figure 4.25.

4.4.5 For the wavelet transform shown in Figure 4.25(b), compute the wavelet

© 2008 by Taylor & Francis Group, LLC

difference reduction encoding (using symbols $+, -, 0, 1$) for the Significance Pass and Refinement Pass when the threshold is 8.

4.4.6$^{\mathbf{c}}$ Produce figures like Figure 4.10 for the image `Barb.bmp`.

4.4.7$^{\mathbf{c}}$ Produce figures like Figure 4.10 for the image `Airfield.bmp`.

4.4.8$^{\mathbf{c}}$ Verify all entries in Table 4.5.

4.4.9$^{\mathbf{c}}$ Add a new part to Table 4.5, labelled FINGERPRINT, by computing PSNR values for each of the three compression ratios using each of the four compression methods for the `fingerprint 1.bmp` image.

4.4.10$^{\mathbf{c}}$ Add a new part to Table 4.5, labelled PEPPERS, by computing PSNR values for each of the three compression ratios using each of the four compression methods for the `peppers.bmp` image.

4.4.11$_{\mathbf{s}}$ Suppose that the symbols $\{a, b, c, d\}$ occur with the frequencies

$$14/21, 5/21, 1/21, 1/21,$$

respectively. Find a Huffman coding for these symbols. What is its average length? What is the minimum theoretical length (the one essentially obtainable by arithmetic coding)?

4.4.12 Suppose that the symbols $\{a, b, c, d\}$ occur with the frequencies

$$1/3, 1/3, 1/6, 1/6,$$

respectively. Find a Huffman coding for these symbols. What is its average length? What is the minimum theoretical length (the one essentially obtainable by arithmetic coding)?

4.4.13 Suppose that the symbols $\{a, b, c, d\}$ occur with the frequencies

$$20/25, 2/25, 2/25, 1/25,$$

respectively. Find a Huffman coding for these symbols. What is its average length? What is the minimum theoretical length (the one essentially obtainable by arithmetic coding)?

4.4.14 Suppose that the symbols $\{a, b, c, d, e\}$ occur with the frequencies

$$20/41, 18/41, 1/41, 1/41, 1/41,$$

respectively. Find a Huffman coding for these symbols. What is its average length? What is the minimum theoretical length (the one essentially obtainable by arithmetic coding)?

Section 4.5

Example 4.5.A [Figure 4.11] To produce Figure 4.11(a), we performed a rather complex procedure. First, we computed a 2-level Daub 9/7 wavelet transform of the `Boat.pgm` image. We then zoomed in on $\mathbf{v}_2$ by right-clicking on the lower left corner

© 2008 by Taylor & Francis Group, LLC

of the wavelet transform image and selecting *Zoom* (and then clicking on the *Zoom* button). After that first zoom, we then right-clicked on the lower right corner and zoomed once more. The resulting displayed image is $\mathbf{v}_2$. We then right-clicked on it and selected *Clip,* which produced a clipped out image of $\mathbf{v}_2$ only.

In this new window, we then graphed the function `(abs(g1)>=16)` to produce an image where the white pixels are parent values significant at threshold 16 (and the black pixels are insignificant parents). We then select *Graph/Interpolate/Haar* in order to produce a new image that is twice as large in each dimension, and each of the pixels (white and black) gives rise to a 2 by 2 child matrix of either all 1 values (if the parent pixel is white) or all 0 values (if the parent pixel is black). This image displays the locations (in white) of all child values in $\mathbf{v}_1$ whose parent values are significant at threshold 16.

To get an image of only those children who are predicted to be *newly significant* at threshold 16, we return to the first window containing the wavelet transform of the `Boat.pgm` image and clip out the $\mathbf{v}_1$ subimage. This is done by right-clicking on the wavelet transform image and selecting *Restore full image* from the popup menu, then right-clicking on the lower-right corner and zooming once, and then clipping out this zoomed subimage. In the resulting new window, we graph the function `(32 > abs(g1) >= 16)` and then copy this image (right-click on it and select *Copy graph* from the popup menu). Returning to the window containing the child values in $\mathbf{v}_1$ with significant parents, we right-click on its image and select *Paste.* Finally, we graph `g1 g2` and that produces a black and white version of the image in (a) [to change it to the gray and white image displayed in Figure 4.11(a); right-click on it and select *Graph style,* and then select the option *Grey (+/-)*].

To produce Figure 4.11(b) we return to the window with the clipping of $\mathbf{v}_1$ and plot `(16>abs(g1)>= 8)` to produce a black and white image of the new significant values in $\mathbf{v}_1$ [which one can then convert to a gray and white image identical to (b) as we did above for (a)].

To get the percentage of correct predictions of 41.2% given in the caption of Figure 4.11, we proceed as follows. First, open a new window by selecting *File/New 2-dim.* Second, copy and paste the image for (a) into this new window, then copy and paste the image for (b) into the window as well. Select *Analysis/Statistics* and compute statistics for graphs 1 and 2. The fraction of energy of graph 1 divided by the energy of graph 2 yields the required percentage.

For Figures (c) and (d), we did similar work. The only differences were that we performed a 3-level transform, and to clip $\mathbf{h}_3$ we zoomed twice in a row after right-clicking on the lower left corner of the wavelet transform, then once on the upper left corner; while to clip $\mathbf{h}_2$ we zoomed once on the lower left corner, followed by once on the upper left corner.

Example 4.5.B [Figure 4.12] The image used was `Barb.bmp` and we applied a 5-level Daub 9/7 transform for the WDR and ASWDR methods and the `real` mode for J2K. (See Examples 4.4.B and 4.2.C for more details on using ImageCompressor and Image Analyzer for doing these compressions.) The decompressed images were all saved as `.bmp` files for loading into the FAWAV program. Then each of these `.bmp` images, and the original `.bmp` image, were loaded into a 2-dim form in FAWAV. We then zoomed in on the same pixel for each of the four graphs (near the tip of Barb's nose).

© 2008 by Taylor & Francis Group, LLC

Example 4.5.C [Figure 4.13] These images were generated in the same way as the last example, except we zoomed on a different pixel.

4.5.1$^{c}_{s}$ Produce images like the ones in Figure 4.11, but for the `Barb.pgm` image. Calculate the percentages of correct predictors as well.

4.5.2^{c} Produce images like the ones in Figure 4.11 for the `Boat.pgm` image, except this time show children of $\mathbf{h}^2$ as predictors of new significant values for $\mathbf{h}^1$, and children of $\mathbf{v}^3$ as predictors of new significant values for $\mathbf{v}^2$. Calculate the percentage of correct predictors as well.

4.5.3^{c} Produce images like the ones in Figure 4.11, but for the `Goldhill.pgm` image. Calculate the percentages of correct predictors as well.

4.5.4^{c} Produce images like the ones in Figure 4.11, but in this case use the `Barb.pgm` image and show children of $\mathbf{h}^2$ as predictors of new significant values for $\mathbf{h}^1$, and children of $\mathbf{v}^3$ as predictors of new significant values for $\mathbf{v}^2$. Calculate the percentage of correct predictors as well.

Section 4.6

Example 4.6.A In Figure 4.26 we illustrate the progressive reconstruction property of ASWDR. To produce these images you select *New Image Processor* from the *File* menu of FAWAV. You then select *Get image* from the *File* menu of the *Image Processor* window. By loading the image `Airfield.pgm` you get the original (uncompressed) image shown in (a). You then select *Compress* from the main menu, and select *Daub 9/7* for the wavelet and enter 1 for the Bit rate (bpp). Click on the *Go* button to save the image in compressed form at 1 bpp (8:1 compression). From that one file, you can generate each of the decompressions in Figure 4.26. You select *Decompress* from the main menu, and enter 0.25 for the bit rate and then click on the *Go* button and select the compressed file for decompression. In that way, you produce the image shown in (b). The images in (c) and (d) are generated by decompressing this same compressed file, using bit rates of 0.5 and 1, respectively. Because we are only using one compressed file, this illustrates progressive reconstruction.

Example 4.6.B [Table 4.6] The data shown in Table 4.6 were generated in the following way. The percentages for WinZip® were generated by using this program[16] to create an archive of compressions of `Barb.pgm` and `Boat.pgm` and `dog.pgm` and reading off the compression percentages provided by WinZip®. To generate the values for ASWDR we used ImageCompressor. By selecting *Get image* from the file menu of ImageCompressor we loaded in an image file. When we loaded an image, we first recorded its file size (obtained by right-clicking on it and selecting *Properties*). We then selected *Compress* from the ImageCompressor menu and entered 9 for the bit rate (that indicates lossless compression because the original image uses 8 bpp), and we also made sure that the wavelet was Daub 5/3 and that the options *arithmetic* for Symbol Encoding and *ASWDR* for Method were checked. Clicking on the *Go* button, we saved the image in a compressed format. When decompressing we again noted the file size of the compressed file before selecting

[16] WinZip® is available from `http://www.winzip.com/`

© 2008 by Taylor & Francis Group, LLC

(a) Original image

(b) 0.25 bpp (32:1 compression)

(c) 0.5 bpp (16:1 compression)

(d) 1.0 bpp (8:1 compression)

FIGURE 4.26
Illustration of Progressive Reconstruction. Each image in (b) to (d) was computed from a single compressed file (saved at 1.0 bpp). First, (b) is reconstructed, then (c), then (d).

it for decompression. Using the uncompressed and compressed file sizes for each image, we calculated the percentages shown in the ASWDR column of the table.

Example 4.6.C [**Figure 4.14**] The decompressions were generated as follows. The `Airfield.pgm` image was opened using an *Image Processor* window in FAWAV. Then we compressed at 200:1 (0.04 bpp as bit rate) using a 4-level Daub 5/3 transform. After decompressing that compressed file we obtained the image in (b). To obtain the image in (d) we right-clicked on the original image (on the left of the window) and drew a rectangle enclosing the region of interest to be losslessly compressed [the airplane at lower left indicated in (c)]. We then compressed the image again, but with a bit rate of 9 bpp and with 0.99 as the *percentage of bits to ROI*. Upon decompression, the selected region of the original image is reconstructed exactly (while some error remains in the rest of the reconstructed image).

4.6.1$^{\mathbf{c}}_{\mathbf{s}}$ Create images like Figure 4.26 for the `boat.pgm` image.

© 2008 by Taylor & Francis Group, LLC

4.6.2^{c} Create images like Figure 4.26 for the `peppers.pgm` image.

4.6.3^{c} Create images like Figure 4.26 for the `goldhill.pgm` image.

4.6.4^{c} Add entries to Table 4.6 for the images `peppers.pgm`, `zelda.pgm`, and `airfield.pgm`. Which of the images (now six in number) shows the least compression? Why do you think it compresses the least?

4.6.5$^{c}_{s}$ Produce images like in Figure 4.14, but this time select a region of interest that contains the swept-wing aircraft just to the left of center. How much savings in file size do you get over sending a lossless compression of the entire image?

4.6.6^{c} Produce images like in Figure 4.14, but this time use the `dog_head.bmp` image and select a region of interest that contains just the jaw region of the dog. How much savings in file size do you get over sending a lossless compression of the entire image?

4.6.7$^{c}_{s}$ If you have access to the lossless compressor program, WinRAR®,[17] then augment Table 4.6 by adding a column with percentages of compression of the three images using WinRAR®.

Section 4.7

In the following example and exercises for section 4.7, we consider the *rate-distortion* curves (R-D curves) for J2K. An R-D curve is a graph of PSNR versus bit-rate for a given image: e.g., bit-rate in increments of 0.1 bpp along the horizontal and PSNR in dBs along the vertical.

Example 4.7.A In Figure 4.27 we show the R-D curve for the `Goldhill.bmp` image using J2K, and for comparison the R-D curve using ASWDR. To obtain the J2K compressions and PSNRs we used Image Analyzer in conjunction with FAWAV. (See Example 4.2.C and the Note that follows it.) To obtain the ASWDR compressions and PSNRs we used ImageCompressor. (Note: We used 7 levels of Daub 9/7 transform and selected the *ASWDR* and *Arithmetic* coding options, and compressed to a 1 bpp file, which we then decompressed at 0.1, 0.2, . . . , 1.0 bpps, and selected *Error* with *PSNR* option to measure the PSNR for each decompression.)

4.7.1^{c} Graph RD-curves for J2K and ASWDR using the `Airfield.bmp` image.

4.7.2^{c} Graph RD-curves for J2K and ASWDR using the `Barb.bmp` image.

4.7.3$^{c}_{s}$ Graph RD-curves for J2K and ASWDR using the `Boat.bmp` image.

4.7.4^{c} Graph RD-curves for J2K and ASWDR using the `Mountain.bmp` image.

4.7.5^{c} Graph RD-curves for J2K and ASWDR using the `Peppers.bmp` image.

Section 4.8

Example 4.8.A [Figure 4.15] To create the image in (a), the image `Boat.bmp` was loaded into FAWAV. The noisy image in (b) was created as follows: random noise

[17] WinRAR® is available from `http://www.rarlab.com/`

© 2008 by Taylor & Francis Group, LLC

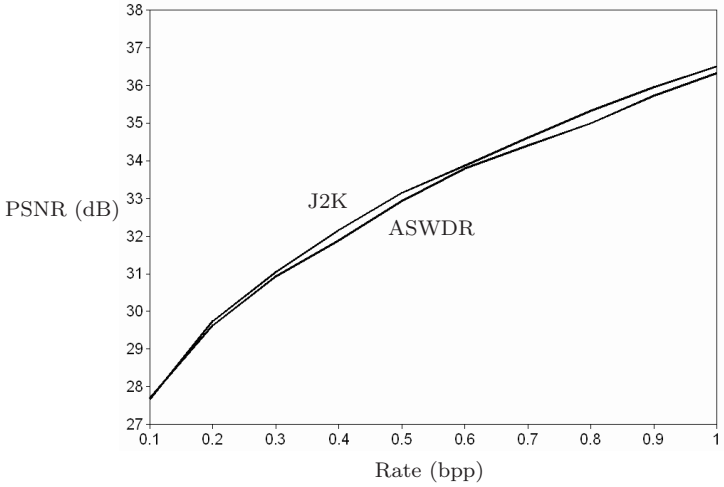


FIGURE 4.27
Rate-Distortion curves for the `Goldhill.bmp` image.

with $\sigma = 20$ was added to the image by plotting `g1+20rang(0)` and then the resulting image was converted to an 8 bpp gray-scale image by selecting *Graph/Quantize (8-bit). After deleting the second graph (the unquantized noisy image),* we created the image in (c) by first performing a 5-level Daub 9/7 wavelet transform of the noisy image, and then plotting the function

```
g3(abs(g3)>= 20sqr(2log(512)))
```

to produce a denoised transform. *After deleting graph 3 (the noisy transform),* we then performed an inverse 5-level Daub 9/7 transform on graph 3 to produce an (unquantized) denoised image. Finally, we deleted graph 3 (the denoised transform) and selected *Graph/Quantize (8-bit)* to create an 8 bpp gray-scale image. This final image is the base threshold denoising shown in (c). To create the image in (d), we deleted the unquantized denoising (graph 3 in the window), and then selected *Graph/Denoise (wavelet).* By clicking the *Plot* button (with graph 2, and 5 levels, and a Daub 9/7 wavelet), we performed a TAWS denoising of the noisy image (which is automatically quantized to an 8-bit gray-scale image), producing the image shown in (d). We also computed PSNRs for each of the images in (b) to (d), in comparison to the original image in (a). We obtained the following results: (b) 22.2 dB, (c) 27.0 dB, (d) 28.8 dB; which show that, for this example, TAWS provides a 1.8 dB improvement over the base threshold method.

Example 4.8.B [Table 4.6] We describe how to obtain the results in the third row, for the *Boats* case with $\sigma = 8$. The other rows of the table are obtained in a similar way. First, we loaded the image `boat.bmp` into FAWAV. We then added noise with $\sigma = 8$ to the image by plotting `g1 + 8rang(0)` and then quantized to 8 bpp (as explained in the previous example). We then deleted graph 2 (the unquantized, noisy image) and saved the noisy gray-scale image to the file:

```
c:\fawave\images\noisy_boat_8.bmp
```

© 2008 by Taylor & Francis Group, LLC

To obtain the Wiener denoising, using MATLAB®, we executed the following three MATLAB® commands:

```
I=imread('c:\fawave\images\noisy_boat_8.bmp');
J=wiener2(I);
imwrite(J,'c:\fawave\images\noisy_boat_8_wiener.png');
```

The last command saves the wiener2 denoised image to a `png` file (a simpler image format for MATLAB® syntax) rather than a `bmp` file. Loading the wiener2 denoised image into FAWAV, we then computed a PSNR value for it in comparison to the original image. That gave the result shown in the table under *Wiener* (your results might differ slightly due to the random nature of the noise). To obtain the results under *TAWS* we performed a TAWS denoising of the noisy image (as described in the previous example), and then computed a PSNR value of the TAWS denoising in comparison to the original image (again, your results might vary slightly due to randomness).

Remark As one can see from this last example, MATLAB® is very simple to operate, and very powerful. Unfortunately, it is also *very expensive* (especially if one has to also buy the IMAGE PROCESSING and SIGNAL PROCESSING toolkits which are needed for the examples described in this book). Because of MATLAB®'s cost, we decided to concentrate on examples using FAWAV (which, although less powerful than MATLAB®, has the advantage of being free).

Example 4.8.C [Figure 4.16] The images in the figure were produced using the method described in the previous example (using the `Barb.bmp` image, and $\sigma = 16$). If, in addition, we zoom in on the upper right corner of all four images, then we obtain the images shown in Figure 4.28.

Example 4.8.D The TAWS denoising in Figure 4.28(d) suffers from some annoying noise residuals (which appear as small pixel-size blemishes). An improved TAWS denoising option is available in FAWAV. It is called TAWS-SPIN and is described in detail in a paper of the author's, *Tree-adapted wavelet shrinkage,* which can be downloaded from

`http://www.uwec.edu/walkerjs/media/TAWSsurv.pdf`

Here we shall illustrate TAWS-SPIN. By selecting the *Denoise (wavelet)* option, and selecting a Daub 9/7 wavelet, you then check the box labeled *Avg.* and select the *2D* option. That performs the TAWS-SPIN algorithm with the parameters recommended in the paper. For instance, performing it on the noisy Barbara image from the previous example, we obtain the image shown in Figure 4.29(d). Compared to Figure 4.28(d), there is a slight improvement in PSNR and the pixel-size blemishes are gone.

Example 4.8.E In Figure 4.30 we compare TAWS-SPIN with wiener2 for the `Boat.bmp` image contaminated with $\sigma = 32$ Gaussian random noise (the noisy image is at the FAWAV webpage as `Noisy_boat_32.bmp`). The TAWS-SPIN denoising shows a much higher PSNR than the wiener2 denoising and has much less noise artifacts.

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image $\sigma = 16$ (24.0 dB)

(c) `wiener2` denoising (27.9 dB)

(d) TAWS denoising (27.9 dB)

FIGURE 4.28
Zooms of two denoisings of Barbara image (PSNRs in parentheses).

Example 4.8.F [Figure 4.18] We loaded the image `STM_Si_111_a.bmp` into FAWAV to produce the image in (a). To denoise it, we performed a 5-level Daub 9/7 transform and then plotted the following function

```
g2 (x<c)(y<c)
+ g2(abs(g2)<350)(1-(x<c)(y<c))(x<a)(y<a)
+ g2 (abs(g1)<t)(1-(x<a)(y<a))
\c = -.5+1/2^5 \a=-.5+1/2^4
\t =20
```

to process the transform. We then performed an inverse 5-level Daub 9/7 transform on the processed transform to produce the denoised image in (d). The images in (b) and (c) were produced by zooming in on portions of the transform of (a), after selecting *Graph style* and choosing the options *LinLog* and *Grey (+/-)*. For (b), we also entered 20 for a threshold.

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image $\sigma = 16$ (24.04 dB)

(c) `wiener2` denoising (27.9 dB)

(d) TAWS-SPIN denoising (28.0 dB)

FIGURE 4.29
Zooms of two denoisings of Barbara image (PSNRs in parentheses).

Example 4.8.G [Figure 4.19] We loaded the image `STM_Si_111_b.bmp` into FAWAV to produce the image in (a). To denoise it, we performed a 5-level Daub 9/7 transform and then plotted the following function

```
g2 (x<c)(y<c) + g2 (abs(g2)<t)(1-(x<c)(y<c))
\c = -.5+1/2^4
\t = 30
```

to process the transform. We then performed an inverse 5-level Daub 9/7 transform on the processed transform to produce the denoised image in (b).

4.8.1[c] Denoise the images `Lena_16.bmp`, `Lena_24.bmp`, `Lena_32.bmp` (noisy versions of `Lena.bmp` with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

© 2008 by Taylor & Francis Group, LLC

(a) Original

(b) Noisy image $\sigma = 32$ (18.2 dB)

(c) `wiener2` denoising (25.0 dB)

(d) TAWS-SPIN denoising (27.4 dB)

FIGURE 4.30
Two denoisings of Boats image (PSNRs in parentheses).

4.8.2$^{\mathbf{c}}$ Denoise the images

`Noisy_Goldhill_16.bmp` `Noisy_Goldhill_24.bmp` `Noisy_Goldhill_32.bmp`

(noisy versions of `Goldhill.bmp` with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.8.3$^{\mathbf{c}}$ Denoise the images

`Noisy_boat_16.bmp` `Noisy_boat_24.bmp` `Noisy_boat_32.bmp`

(noisy versions of `boat.bmp` with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) wiener2. Which method appears best (both ob-

© 2008 by Taylor & Francis Group, LLC

jectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.8.4^{c} Denoise the images

`Noisy_peppers_16.bmp  Noisy_peppers_24.bmp  Noisy_peppers_32.bmp`

(noisy versions of `Peppers.bmp` with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.8.5$^{c}_{s}$ The image `Elaine.bmp` has some noise (try zooming a couple times on the center pixel to see it more clearly). Denoise this image using TAWS, and TAWS-SPIN, and (if available) wiener2. Which method appears best subjectively, in terms of how it appears to you visually?

4.8.6^{c} Try to improve the denoising of the `STM_Si_111_a.bmp` image, obtaining a more sharply focused image than the example given in the text (see Example 4.8.F).

4.8.7^{c} Try to improve the denoising of the `STM_Si_111_b.bmp` image, obtaining a more sharply focused image than the example given in the text (see Example 4.8.G).

Section 4.9

Example 4.9.A [Figure 4.20] To produce (a) we first graphed an octagon shaped figure by plotting the following function

```
(y-x-c<=0)(x+y-c<=0)(x-y-c<=0)
(x+y+c>=0)(abs(y)<=b)(abs(x)<=b)
\c=12/5\b=8.5/5
```

over the region $[-4, 4] \times [-4, 4]$ using 128 as the choice for *Points*. We then computed a 1-level Coif6 transform and plotted

```
10g2(1-(x<0)(y<0))
```

After changing the display style of the resulting graph to *Log*, with *Grey (+/-)*, and setting a threshold of 0.1, we obtained the image shown in (a). To obtain the image in (b) we then performed an inverse 1-level Coif6 transform on image (a) and changed its graph style to *Grey (+/-)*.

Example 4.9.B [Figure 4.21] To obtain the images in (a) and (b) we first loaded the image `house.pgm` and that gave us (a). We then performed a 1-level Daub4 transform on this image and plotted the function

```
g2+2g2(1-(x<0)(y<0))
```

to obtain a processed transform. We then performed an inverse 1-level Daub4 transform of this processed transform. By quantizing to get an 8-bit gray-scale image, we obtained the image (b).

© 2008 by Taylor & Francis Group, LLC

Example 4.9.C [Figure 4.22] The image in (a) was obtained by successively loading the images `goldhill.pgm`, `boat.pgm`, `airfield.pgm`, `peppers.pgm` into a 2-dim form. The image in (b) was obtained by performing 2-level Coif18 transforms of these images, copying and pasting them into a single 2-dim form, and changing their graph styles to *Lin-Log* 128 and choosing a threshold of 4.

Example 4.9.D [Table 4.7] We explain how the entries for the column labeled *Second* were obtained (the entries for the other columns were obtained in a similar way). First we obtained the denoised image of the noisy boats image (see Example 4.8.A). We copied and pasted this denoised image into a new 2-dim form. We then produced a 2-level Coif18 transform of this denoised image, and displayed it with the same graph style as the images in Figure 4.22(b) discussed in Example 4.9.C. The next step is to compute the relative 2-norm differences between the trend subimages. To do that for the Gr 1 entry (corresponding to the Goldhill image) we zoomed in on the lower left corner of the Coif18 transform of the Goldhill image, so that just the trend subimage was displayed. We then clipped this subimage to obtain a new 2-dim form containing a display of just the trend subimage, and repeated this process with the denoised Boats image transform. We then copied the trend subimage from the Goldhill image and pasted it into the form containing the trend subimage from the denoised Boats image. Finally, we computed a 2-norm difference between graphs 1 and 2, with the option *Relative* selected. That gave us the value 0.455 shown in the table. We then repeated this process with the other images to get the remaining entries. *Note:* The values you obtain may differ slightly from the ones reported in Table 4.7 due to the random nature of the additive noise.

4.9.1^{c} Produce an image of the edges of `Peppers.bmp`.

4.9.2$^{c}_{s}$ Edge enhance the image `cathedra.pgm`.

4.9.3^{c} Edge enhance the image `Peppers.bmp`.

4.9.4^{c} Add additional data to Table 4.7, using the image `Elaine.pgm`.

4.9.5^{c} Add additional data to Table 4.7, using the image `Zelda.pgm`.

4.9.6$^{c}_{s}$ Construct a table analogous to Table 4.7, using 32:1 compressions of `Lena.pgm` compared to the images `Barb.bmp`, `Zelda.bmp`, `Lena.pgm`, `Peppers.bmp`.

© 2008 by Taylor & Francis Group, LLC

Chapter 5

Frequency analysis

If our ear uses a certain technique to analyze a signal, then if you use that same mathematical technique, you will be doing something like our ear. You might miss important things, but you would miss things that our ear would miss too.

—Ingrid Daubechies[1]

It is well known that human hearing responds to the frequencies of sound. Evolution has shaped our sense of hearing into a superb frequency analyzer, whose full characteristics we do not yet completely understand. Nevertheless, it is clear that we perceive tones of different frequency as having different pitch, and musical notes are called higher or lower depending on whether they have a corresponding higher or lower frequency.

The mathematical analysis of the frequency content of signals is called *Fourier analysis.* In this chapter we shall sketch the basic outlines of Fourier analysis as it applies to discrete signals and use it to analyze the frequency content of wavelets. A deeper understanding of wavelets can be gained from studying their frequency content, and by examining how this frequency content relates to wavelet transforms of signals. To keep the mathematics as simple as possible we shall focus on 1D signals, although in Section 5.5 we shall describe some 2D applications. We shall also examine a close cousin of wavelet transforms, the *Gabor transforms.* Gabor transforms provide a dynamic frequency analysis of signals that is a useful alternative to wavelet

[1] Daubechies' quote is from [1].

© 2008 by Taylor & Francis Group, LLC

methods, particularly in the field of music. We will show how Gabor transforms can be used for musical analysis, musical synthesis, and denoising.

5.1 Discrete Fourier analysis

The frequency content of a discrete signal is revealed by its *discrete Fourier transform* (DFT). The DFT of a discrete signal is usually performed on a computer using an algorithm called a *fast Fourier transform* (FFT). Our focus in this primer is not to prove the results about the DFT that we shall be using (see, however, Exercises 5.2.1 through 5.2.4). Our emphasis instead will be on how the DFT is applied in signal analysis, especially in wavelet analysis. There are many excellent treatments of the DFT and the FFT available for the reader who desires further discussion and proofs; we list a few of these references at the end of this chapter. In any case, the results we shall be using are all standard ones that are quite well known. Our approach will be to illustrate some fundamental ideas via a few examples rather than via a theoretical derivation.

The DFT reveals the frequency content of a discrete signal; so we shall begin by reviewing the notion of frequency. The analog signals $\cos 2\pi\nu x$ and $\sin 2\pi\nu x$, where x denotes time, both have a fundamental time period of $1/\nu$. Consequently, these signals repeat their basic form ν times in a unit-length time interval, i.e., they have a *frequency* of ν cycles/unit-time. The notion of frequency of a discrete signal is closely related to frequency of analog signals, as the following examples show. In the next section we shall make this connection more precise, but it may help to first examine some illustrative examples.

As a first example of a DFT, consider the discrete signal, Signal I, shown in Figure 5.1(a). Signal I consists of 1024 samples of the analog signal

$$g_1(x) = 2\cos 4\pi x + 0.5\sin 24\pi x. \tag{5.1}$$

It has frequencies of 2 and 12 for its two terms. In Figure 5.1(b) we show the DFT of Signal I. The two inner spikes are located at ± 2, corresponding to the frequency 2 of the cosine term of Signal I, and the two outer spikes are located at ± 12, corresponding to the frequency 12 of the sine term of Signal I. Because of the way in which the DFT is defined (see the next section), the spikes appear at $\pm\nu$ instead of just at ν when the signal contains a sine or cosine term of frequency ν. Notice that the lengths of the spikes in the DFT of Signal I that correspond to the cosine term in Signal I appear to be about 4 times greater than the lengths of the spikes that correspond to the sine term, and 4 is also the ratio of the amplitude of the cosine term to the amplitude of the sine term in Signal I. This example shows how the DFT can identify the frequencies—and the relative sizes of their contributions in terms of amplitude—that make up the frequency content of a signal.

© 2008 by Taylor & Francis Group, LLC

As a second example of a DFT, consider the discrete signal, Signal II, shown in Figure 5.1(c). Signal II consists of 1024 samples of the analog signal

$$g_2(x) = \frac{1 + \cos 24\pi x}{1 + 4x^2} \,. \tag{5.2}$$

Its DFT is shown in Figure 5.1(d). The significant values in the DFT are clustered around 0 and ±12. These values of ±12 clearly correspond to the frequency 12 in the cosine term in Formula (5.2), but the cause of the cluster of significant DFT values around 0 is less clear. We shall see what these very low frequency values have to do with Signal II in Section 5.3, when we examine the frequency content of averaged signals in wavelet multiresolution analysis.

5.1.1 Frequency content of wavelets

As a third example of DFTs, we consider the frequency content of scaling signals and wavelets. This will enable us to comprehend the effects of wavelet

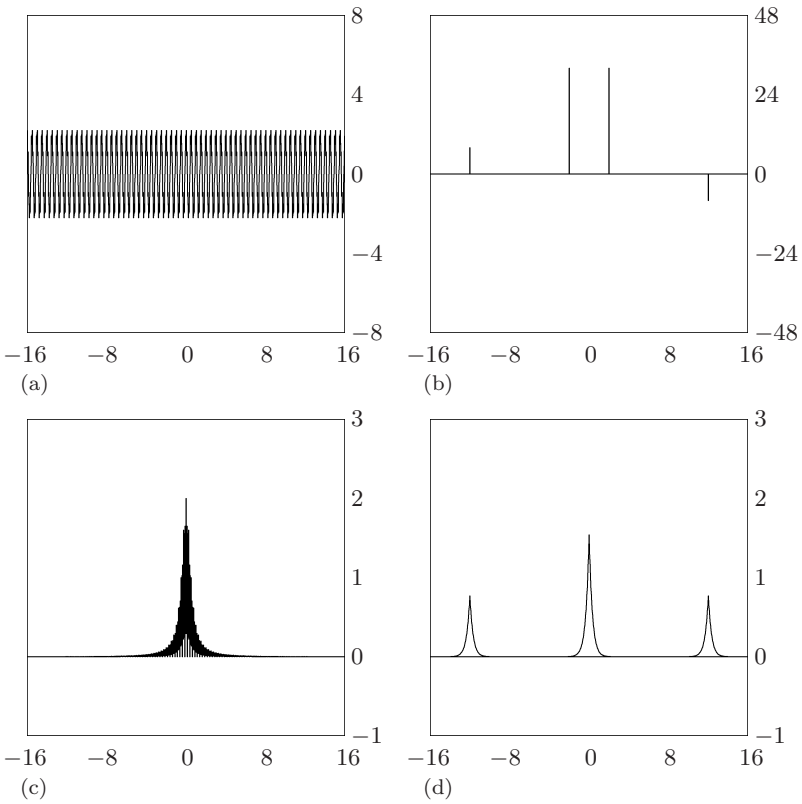


FIGURE 5.1
(a) Signal I. (b) DFT of Signal I. (c) Signal II. (d) DFT of Signal II.

© 2008 by Taylor & Francis Group, LLC

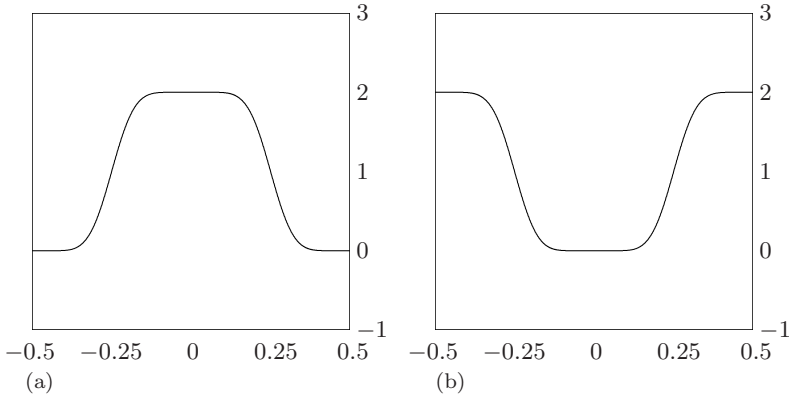


FIGURE 5.2
(a) Squares of magnitudes of DFT of Coif12 scaling signal $\mathbf{V}_1^1$, its spectrum. (b) Spectrum of Coif12 wavelet $\mathbf{W}_1^1$.

MRAs on the frequency content of a signal, which we shall examine in Section 5.3.

As a typical case of scaling signals and wavelets, consider the Coif12 scaling signal $\mathbf{V}_1^1$ and wavelet $\mathbf{W}_1^1$. In Figure 5.2(a) we show the squares of the magnitudes of the DFT of the Coif12 scaling signal $\mathbf{V}_1^1$, which is called the *spectrum* of this signal. Notice that the significant, non-zero values of this spectrum are concentrated in the central half of the graph near the origin, which are the lower frequencies. The higher frequencies lie in the left and right quarters of the graph, and for these higher frequencies the values of the spectrum are approximately zero. In contrast, consider the spectrum of the wavelet $\mathbf{W}_1^1$, shown in Figure 5.2(b). Its significant, non-zero values are concentrated in the higher frequencies, while for the lower frequencies its values are approximately zero.

Notice that the graphs in Figure 5.2 seem complementary to one another. In fact, it is the case that the sum of their values equals the constant 2. This is a fundamental property of the spectrums of first-level scaling signals and wavelets.

5.2 Definition of the DFT and its properties

In this section, we shall state the definition of the DFT and some of its principal properties. We shall be fairly brief; more thorough discussions can be found in many references, some of which are listed at the end of this chapter.

To state the definition of the DFT in a succinct form, we shall make use of the Σ-notation for summation. In this notation, a sum $g_1 + g_2 + \cdots + g_M$

© 2008 by Taylor & Francis Group, LLC

is expressed as

$$\sum_{m=1}^{M} g_m.$$

For instance, the following sum

$$a + ar + ar^2 + \cdots + ar^{M-1} = \sum_{m=1}^{M} ar^{m-1}$$

should be familiar from algebra.

Using this summation notation, we can state the definition of the DFT of a signal $\mathbf{f}$ of length N. We shall denote this DFT by $\mathcal{F}\mathbf{f}$, and its values $(\mathcal{F}\mathbf{f})_n$ are defined by

$$(\mathcal{F}\mathbf{f})_n = \sum_{m=1}^{N} f_m e^{-i2\pi(n-1)(m-1)/N}. \tag{5.3}$$

Although the variable n in (5.3) can be any integer, we shall see that the periodicity property stated below implies that the values $(\mathcal{F}\mathbf{f})_n$ for $n = -N/2$ to $n = N/2 - 1$ are sufficient for describing $\mathcal{F}\mathbf{f}$.

In Formula (5.3) we use the complex exponentials $e^{-i2\pi(n-1)(m-1)/N}$, defined via *Euler's formulas:*

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{5.4}$$

and

$$e^{-i\theta} = \cos\theta - i\sin\theta. \tag{5.5}$$

One consequence of (5.5) is that the modulus (or magnitude), $\left|e^{-i\theta}\right|$, of the complex number $e^{-i\theta}$ is equal to 1.

Another consequence of (5.4) and (5.5) is that

$$\cos 2\pi\nu x = \frac{1}{2}\, e^{i2\pi\nu x} + \frac{1}{2}\, e^{-i2\pi\nu x}. \tag{5.6}$$

This equation shows that $\cos 2\pi\nu x$ can be expressed as a sum of two complex exponentials having frequencies $\pm\nu$, which should remind the reader of the DFT of Signal I discussed in the previous section. Similarly,

$$\sin 2\pi\nu x = \frac{1}{2i}\, e^{i2\pi\nu x} - \frac{1}{2i}\, e^{-i2\pi\nu x} \tag{5.7}$$

which shows that $\sin 2\pi\nu x$ can also be expressed as a sum of complex exponentials having frequencies of $\pm\nu$.

5.2.1 Properties of the DFT

The following four properties are the principal ones satisfied by the DFT. We use the notation $\mathbf{f} \overset{\mathcal{F}}{\longmapsto} \mathcal{F}\mathbf{f}$ to symbolize the DFT operation.

© 2008 by Taylor & Francis Group, LLC

1. *Linearity.* For all constants α and β, and all signals $\mathbf{f}$ and $\mathbf{g}$ of length N,

$$\alpha\mathbf{f} + \beta\mathbf{g} \stackrel{\mathcal{F}}{\longmapsto} \alpha\mathcal{F}\mathbf{f} + \beta\mathcal{F}\mathbf{g}.$$

2. *Periodicity.* If $\mathbf{f}$ is a signal of length N, then $\mathcal{F}\mathbf{f}$ has period N; that is,

$$(\mathcal{F}\mathbf{f})_{n+N} = (\mathcal{F}\mathbf{f})_n \tag{5.8}$$

holds for all integers n.

3. *Inversion.* The signal $\mathbf{f}$ can be obtained from $\mathcal{F}\mathbf{f}$ by

$$f_m = \frac{1}{N} \sum_{n=-N/2}^{N/2-1} (\mathcal{F}\mathbf{f})_n \, e^{i2\pi(n-1)(m-1)/N} \tag{5.9}$$

for $m = 1, 2, \ldots, N$.

4. *Parseval's Equality.* The signal $\mathbf{f}$ and its DFT $\mathcal{F}\mathbf{f}$ satisfy

$$\sum_{m=1}^{N} |f_m|^2 = \frac{1}{N} \sum_{n=-N/2}^{N/2-1} |(\mathcal{F}\mathbf{f})_n|^2 . \tag{5.10}$$

Because of periodicity, the N values $(\mathcal{F}\mathbf{f})_n$ for $n = -N/2$ to $n = N/2 - 1$ are sufficient for uniquely determining a DFT $\mathcal{F}\mathbf{f}$. It is for this reason that all of our figures of DFTs have the same number of values as the signal being transformed; and all software packages that compute DFTs follow this same convention.

The Inversion Formula (5.9) is particularly important. For one thing, it ensures that distinct signals must have distinct DFTs. For another, it allows for a useful interpretation of DFT values. Each DFT value $(\mathcal{F}\mathbf{f})_n$ is an amplitude for a discrete complex exponential signal

$$e^{i2\pi(n-1)(m-1)/N}, \quad m = 1, 2, \ldots, N, \tag{5.11}$$

which is a sampled version of $e^{i2\pi(n-1)x}$, a complex exponential analog signal of frequency $n - 1$. The sample points are $x_m = (m-1)/N$. Or, since

$$\frac{(n-1)(m-1)}{N} = \frac{n-1}{\Omega} \cdot \frac{(m-1)\Omega}{N},$$

one can also view (5.11) as sample values of $e^{i2\pi(n-1)x/\Omega}$, a complex exponential of frequency $(n-1)/\Omega$. In this latter case, the sample points are $x_m = (m-1)\Omega/N$. This latter case is important when signals are obtained from measured samples of analog signals over a time interval of length Ω. In any case, the Inversion Formula (5.9) shows that the signal $\mathbf{f}$ can be realized by summing these discrete complex exponential signals with amplitudes given by the DFT values $(\mathcal{F}\mathbf{f})_n$, and multiplying by the scale factor $1/N$.

© 2008 by Taylor & Francis Group, LLC

Parseval's Equality (5.10) can be regarded as a conservation of energy property, provided we include a scale factor of $1/N$ again. This conservation of energy, as with wavelet transforms, is useful for understanding how to make applications of the DFT operation. In fact, in some instances, the DFT, or closely related transforms like the Gabor transforms, can be used for compression and noise removal. We shall briefly explore these ideas later in this chapter.

One final remark needs to be made about periodicity and inversion. They imply, since the right side of (5.9) also has period N in the integer variable m, that the finite signal $\mathbf{f}$ should be assumed to be a subsignal of a periodic signal having period N. That is,

$$f_{m+N} = f_m \tag{5.12}$$

for all integers m. Notice that (5.12) is the wrap-around property of signals that we made use of for scaling signals and wavelets in Chapter 3.

5.2.2 z-transforms *

The z-transform provides a more flexible way of expressing the values of the DFT, which is especially helpful in wavelet theory. Since some readers may find the theory of z-transforms to be difficult, we shall treat it as optional material. This material will be used later only in optional sections.

The z-transform of $\mathbf{f}$ will be denoted by $\mathbf{f}[z]$ and is defined by

$$\mathbf{f}[z] = \sum_{m=1}^{N} f_m z^{m-1}. \tag{5.13}$$

The variable z takes its values on the unit-circle of the complex plane.[2]

If we set z equal to $e^{-i2\pi(n-1)/N}$, then

$$\mathbf{f}[e^{-i2\pi(n-1)/N}] = (\mathcal{F}\mathbf{f})_n. \tag{5.14}$$

Formula (5.14) shows that the DFT, $\mathcal{F}\mathbf{f}$, consists of the values of $\mathbf{f}[z]$ at the points $z = e^{-i2\pi(n-1)/N}$ which lie uniformly spaced around the unit-circle in the complex plane.

One application of the z-transform is to the computation of DFTs of scaling signals and wavelets. To do this, we must first define the *cyclic translation* of a signal $\mathbf{f}$. The cyclic translation forward by 1 unit is denoted by $\mathcal{T}_1\mathbf{f}$ and is defined by

$$\mathcal{T}_1\mathbf{f} = (f_N, f_1, f_2, \ldots, f_{N-1}). \tag{5.15}$$

Notice the wrap-around at the end of $\mathcal{T}_1\mathbf{f}$; because of this wrap-around we use the adjective *cyclic* in describing this translation. The cyclic translation

[2] That is, $|z| = 1$. We also note that there are other definitions of the z-transform. The one we are working with is most useful for finite length signals.

© 2008 by Taylor & Francis Group, LLC

backward by 1 unit is denoted by $\mathcal{T}_{-1}\mathbf{f}$ and is defined by

$$\mathcal{T}_{-1}\mathbf{f} = (f_2, f_3, \ldots, f_N, f_1). \tag{5.16}$$

All other cyclic translations are defined via compositions, e.g., $\mathcal{T}_2 = \mathcal{T}_1 \circ \mathcal{T}_1$, $\mathcal{T}_3 = \mathcal{T}_1 \circ \mathcal{T}_1 \circ \mathcal{T}_1$, and so on. The most important property of these cyclic translations is that $\mathcal{T}_k \circ \mathcal{T}_m = \mathcal{T}_{k+m}$ for all integers k and m.

The key property of the z-transform is that

$$\mathcal{T}_k\mathbf{f}[z] = \mathbf{f}[z]z^k \tag{5.17}$$

which holds *provided z is equal to $e^{-i2\pi(n-1)/N}$ for some integer n.* To see why (5.17) holds, we demonstrate it for $k = 1$, and higher powers of k follow by repeating the argument for $k = 1$. If $k = 1$, then

$$\mathcal{T}_1\mathbf{f}[z] = f_N + f_1 z + \cdots + f_{N-1}z^{N-1}.$$

But, if $z = e^{-i2\pi(n-1)/N}$, then $z^N = 1$. Consequently

$$\begin{aligned}\mathcal{T}_1\mathbf{f}[z] &= f_1 z + \cdots + f_{N-1}z^{N-1} + f_N z^N \\ &= (f_1 + f_2 z + \cdots + f_N z^{N-1})z \\ &= \mathbf{f}[z]z\end{aligned}$$

which proves (5.17) for $k = 1$.

One application of (5.17) is to the relationship between the frequency content of $\mathbf{V}_1^1$ and all the other first-level scaling signals. Since $\mathbf{V}_m^1 = \mathcal{T}_{2m-2}\mathbf{V}_1^1$, it follows from (5.17) that

$$\mathbf{V}_m^1[z] = \mathbf{V}_1^1[z]z^{2m-2}. \tag{5.18}$$

Similarly, we have

$$\mathbf{W}_m^1[z] = \mathbf{W}_1^1[z]z^{2m-2} \tag{5.19}$$

for the first-level wavelets. These last two formulas clarify the relationship between the frequency contents of scaling signals and wavelets because Formula (5.14) relates z-transforms to DFTs. For example, because $|z| = 1$, it follows that

$$\left|\mathcal{F}\mathbf{V}_m^1\right|^2 = \left|\mathcal{F}\mathbf{V}_1^1\right|^2, \quad \left|\mathcal{F}\mathbf{W}_m^1\right|^2 = \left|\mathcal{F}\mathbf{W}_1^1\right|^2$$

which shows that the spectrum of each first-level scaling signal is equal to the spectrum of $\mathbf{V}_1^1$ and the spectrum of each first-level wavelet is equal to the spectrum of $\mathbf{W}_1^1$. We shall make further use of Formulas (5.18) and (5.19) in the next section.

5.3 Frequency description of wavelet analysis

In this section we shall examine how the frequency content of a signal is changed when the signal undergoes a wavelet analysis. To be more precise,

© 2008 by Taylor & Francis Group, LLC

we shall compare the frequency contents of the averaged signals $\mathbf{A}^k$ and the detail signals $\mathbf{D}^k$ created in a k-level MRA with the frequency content of the original signal $\mathbf{f}$.

Let's examine a 1-level Coif12 MRA of Signal II, shown in Figure 5.1(c). This example illustrates the fundamental aspects of the effect on frequency content of an MRA using a Daubechies wavelet. At the end of this section we shall discuss the mathematical details in complete generality.

When a first averaged signal $\mathbf{A}^1$ is created, it consists of a sum of multiples of the first level scaling signals [see Formula (3.13b) on p. 47]. In Figure 5.3(a) we show the first averaged signal, using Coif12 scaling signals, for Signal II. The DFT of this first averaged signal is shown in Figure 5.3(b). It is interesting to compare this DFT with the DFT of Signal II in Figure 5.1(d) and the spectrum of the Coif12 scaling signal $\mathbf{V}_1^1$ in Figure 5.2(a). In order to make this comparison, the values of the scaling signal spectrum must be graphed over $[-16, 16]$ instead of $[-0.5, 0.5]$, but this can be easily done by a change of scale of the horizontal (frequency) axis. The values of this scaling signal spectrum are approximately equal to the constant 2 near the origin (for the lower frequencies), and are approximately equal to the constant 0 at the left and right (for the higher frequencies).

The relation between the DFT of $\mathbf{A}^1$ and the spectrum $\left|\mathcal{F}\mathbf{V}_1^1\right|^2$ of $\mathbf{V}_1^1$ is

$$\mathcal{F}\mathbf{A}^1 \approx \frac{1}{2}\left|\mathcal{F}\mathbf{V}_1^1\right|^2 \mathcal{F}\mathbf{f}. \tag{5.20}$$

On the right side of (5.20) is the product of the two signals $\left|\mathcal{F}\mathbf{V}_1^1\right|^2$ and $\mathcal{F}\mathbf{f}$; hence this approximation says that each value $(\mathcal{F}\mathbf{A}^1)_n$ of the DFT of $\mathbf{A}^1$ is approximately equal to 1/2 times $\left|(\mathcal{F}\mathbf{V}_1^1)_n\right|^2 (\mathcal{F}\mathbf{f})_n$. Thus 1/2 times the spectrum of $\mathbf{V}_1^1$ acts as a *low-pass filter* on the values of the DFT of $\mathbf{f}$, allowing through only the low frequency values (since $\left|\mathcal{F}\mathbf{V}_1^1\right|^2 \approx 0$ for high frequency values).

It is interesting to examine the way in which this low-pass filtering affects the averaged signal. By comparing the averaged signal $\mathbf{A}^1$ in Figure 5.3(a) with the original signal $\mathbf{f}$ in Figure 5.1(c) we can see that there is much less rapid oscillation in the averaged signal; this is due to the suppression of the high frequencies by the low-pass filtering.

In contrast to the scaling signal, 1/2 times the spectrum of the Coif12 wavelet $\mathbf{W}_1^1$ acts as a *high-pass filter,* allowing through only the high frequency portions of the DFT of $\mathbf{f}$. In fact, the following approximation holds

$$\mathcal{F}\mathbf{D}^1 \approx \frac{1}{2}\left|\mathcal{F}\mathbf{W}_1^1\right|^2 \mathcal{F}\mathbf{f}. \tag{5.21}$$

As can be seen from Figure 5.2(b), the factor $\left|\mathcal{F}\mathbf{W}_1^1\right|^2$ is approximately zero for the low frequency values, and is approximately 2 for the high frequency values. It then follows from (5.21)—and we can check it by examining Figure 5.3(d)—that the DFT of the first detail signal mostly consists of the higher frequency

© 2008 by Taylor & Francis Group, LLC

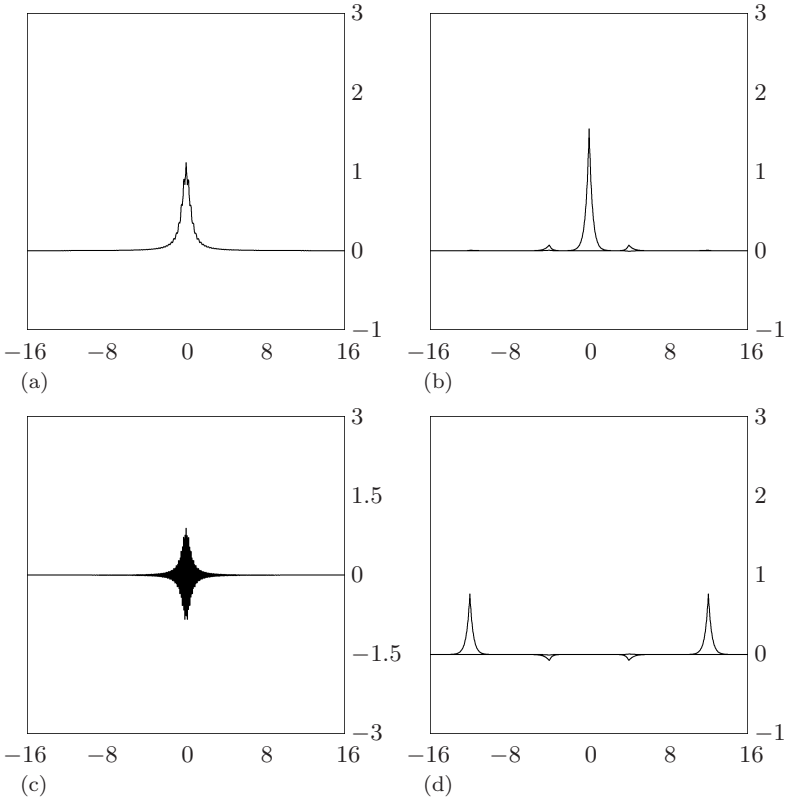


FIGURE 5.3
Frequency analysis of 1-level Coif12 MRA. (a) First averaged signal $\mathbf{A}^1$ for Signal II [c.f. Figure 5.1(c)]. (b) DFT of first averaged signal [c.f. Figures 5.1(d) and 5.2(a)]. (c) First detail signal $\mathbf{D}^1$ for Signal II. (d) DFT of first detail signal [c.f. Figures 5.1(d) and 5.2(b)].

values of the DFT of the signal $\mathbf{f}$. The effect that this has on the detail signal $\mathbf{D}^1$ is that it contains the most rapid, high frequency oscillations from the original signal.

This example of a 1-level Coif12 MRA of Signal II is typical of all wavelet MRAs. The frequency content of the first averaged signal $\mathbf{A}^1$ consists of low frequency values resulting from a low-pass filtering of the frequency content of the signal by 1/2 times the spectrum of the 1-level scaling signal $\mathbf{V}_1^1$. And the first-level detail signal $\mathbf{D}^1$ has a frequency content obtained from a high-pass filtering of the frequency content of the signal by 1/2 times the spectrum of the 1-level wavelet $\mathbf{W}_1^1$.

As with the first level, the DFTs of higher level averaged signals and fluctuation signals can be obtained by multiplying the signal's DFT by the spectra of higher level scaling signals and wavelets. For example, the DFT of

© 2008 by Taylor & Francis Group, LLC

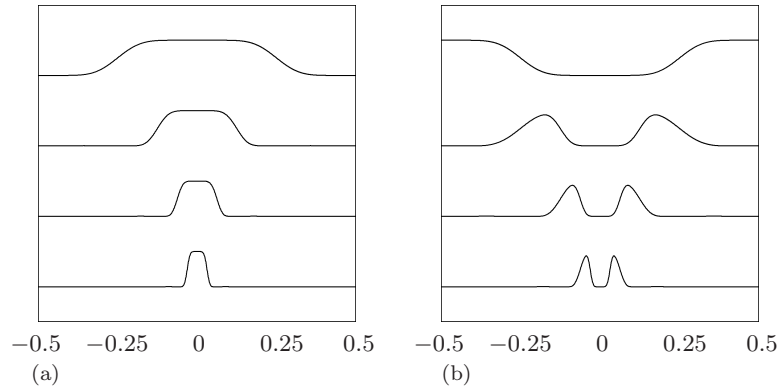


FIGURE 5.4
(a) Spectra of Coif12 scaling signals, from level $k = 1$ at top to level $k = 4$ at bottom. (b) Spectra of Coif12 wavelets from level $k = 1$ at top to level $k = 4$ at bottom. The spectra have been multiplied by constants 2^{-k} for each k in order to make their graphs of similar size.

the second averaged signal $\mathbf{A}^2$ satisfies

$$\mathcal{F}\mathbf{A}^2 \approx \frac{1}{4}\left|\mathcal{F}\mathbf{V}_1^2\right|^2 \mathcal{F}\mathbf{f} \tag{5.22}$$

and the DFT of the second detail signal $\mathbf{D}^2$ satisfies

$$\mathcal{F}\mathbf{D}^2 \approx \frac{1}{4}\left|\mathcal{F}\mathbf{W}_1^2\right|^2 \mathcal{F}\mathbf{f}. \tag{5.23}$$

In Formula (5.22), 1/4 times the spectrum $\left|\mathcal{F}\mathbf{V}_1^2\right|^2$ of the 2-level scaling signal $\mathbf{V}_1^2$ acts as a low-pass filter. This low-pass filter is graphed in Figure 5.4(a) as the second graph from the top. Notice that this low-pass filter allows through only much lower frequencies than the first-level filtering due to $\left|\mathcal{F}\mathbf{V}_1^1\right|^2$. The spectrum $\left|\mathcal{F}\mathbf{W}_1^2\right|^2$ in Formula (5.23), however, does not act as a high-pass filter. As can be seen from the second graph at the top of Figure 5.4(b), 1/4 times $\left|\mathcal{F}\mathbf{W}_1^2\right|^2$ acts as a *band-pass* filter, in the sense that only two small isolated bands of frequency values are non-zero for this spectrum.

Higher level MRAs follow this same pattern. The k-level averaged signal $\mathbf{A}^k$ has a DFT which satisfies

$$\mathcal{F}\mathbf{A}^k \approx \frac{1}{2^k}\left|\mathcal{F}\mathbf{V}_1^k\right|^2 \mathcal{F}\mathbf{f}. \tag{5.24}$$

And the signal $\mathbf{L}_k = 2^{-k}\left|\mathcal{F}\mathbf{V}_1^k\right|^2$ acts as a low-pass filter. In Figure 5.4(a) we show these low-pass filters $\mathbf{L}_k$ for $k = 3$ and $k = 4$. Notice that only a very small interval of values around the origin are non-zero for $\mathbf{L}_4$. The k-level detail signal's DFT satisfies

$$\mathcal{F}\mathbf{D}^k \approx \frac{1}{2^k}\left|\mathcal{F}\mathbf{W}_1^k\right|^2 \mathcal{F}\mathbf{f}. \tag{5.25}$$

© 2008 by Taylor & Francis Group, LLC

As can be seen from Figure 5.4(b), for $k > 1$, the signal $2^{-k}\left|\mathcal{F}\mathbf{W}_1^k\right|^2$ acts as a band-pass filter. The bands of non-zero values for this filter consist of the values lying between the downward sloping left and right sides of the central portions of the graphs of the low-pass filters $\mathbf{L}_k$ and $\mathbf{L}_{k-1}$.

5.3.1 Low-pass and high-pass filtering *

In this optional subsection we shall discuss the mathematical formulation of the low-pass and high-pass filtering interpretation of wavelet MRAs. Our tool will be the z-transform described at the end of Section 5.2.

From Formula (3.13b) on p. 47, we have

$$\mathbf{A}^1 = a_1\mathbf{V}_1^1 + a_2\mathbf{V}_2^1 + \cdots + a_{N/2}\mathbf{V}_{N/2}^1 \tag{5.26}$$

where $(a_1, a_2, \ldots, a_{N/2})$ is the first trend subsignal $\mathbf{a}^1$. Formula (5.14) tells us that to obtain the DFT, $\mathcal{F}\mathbf{A}^1$, of $\mathbf{A}^1$ it suffices to obtain its z-transform $\mathbf{A}^1[z]$. By Formulas (5.26) and (5.18) we have

$$\begin{aligned}\mathbf{A}^1[z] &= a_1\mathbf{V}_1^1[z] + a_2\mathbf{V}_2^1[z] + a_3\mathbf{V}_3^1[z] + \cdots + a_{N/2}\mathbf{V}_{N/2}^1[z]\\ &= a_1\mathbf{V}_1^1[z] + a_2\mathbf{V}_1^1[z]z^2 + a_3\mathbf{V}_1^1[z]z^4 + \cdots + a_{N/2}\mathbf{V}_1^1[z]z^{N-2}.\end{aligned}$$

Thus $\mathbf{A}^1[z]$ satisfies

$$\mathbf{A}^1[z] = \mathbf{V}_1^1[z]\left(a_1 + a_2z^2 + a_3z^4 + \cdots + a_{N/2}z^{N-2}\right). \tag{5.27}$$

Because of Formula (5.27) we see that the DFT of $\mathbf{A}^1$ is obtained as a product of the DFT of $\mathbf{V}_1^1$ with the DFT obtained from the polynomial

$$a_1 + a_2z^2 + a_3z^4 + \cdots + a_{N/2}z^{N-2} = \mathbf{a}^1[z^2].$$

It now remains to examine the connection between the polynomial $\mathbf{a}^1[z^2]$ and the z-transform of the signal $\mathbf{f}$.

The polynomial $\mathbf{a}^1[z^2]$ satisfies

$$\begin{aligned}\mathbf{a}^1[z^2] &= a_1 + a_2z^2 + a_3z^4 + \cdots + a_{N/2}z^{N-2}\\ &= (\mathbf{f}\cdot\mathbf{V}_1^1) + (\mathbf{f}\cdot\mathbf{V}_2^1)z^2 + (\mathbf{f}\cdot\mathbf{V}_3^1)z^4 + \cdots + (\mathbf{f}\cdot\mathbf{V}_{N/2}^1)z^{N-2}\\ &= \sum_{m=1}^{N/2}(\mathbf{f}\cdot\mathbf{V}_m^1)z^{2m-2}\\ &= \sum_{m=1}^{N/2}(\mathbf{f}\cdot\mathcal{T}_{2m-2}\mathbf{V}_1^1)z^{2m-2}.\end{aligned} \tag{5.28}$$

This last polynomial in z consists of the even powered terms from the polynomial on the left side of the following identity:

$$\sum_{k=1}^{N}(\mathbf{f}\cdot\mathcal{T}_{k-1}\mathbf{V}_1^1)z^{k-1} = \mathbf{f}[z]\mathbf{V}_1^1[z^{-1}]. \tag{5.29}$$

© 2008 by Taylor & Francis Group, LLC

The fact that (5.29) holds is a consequence of the definition of multiplication of polynomials; we leave its proof as an exercise for the reader.

The sum of even powered terms can be extracted from the polynomial on the right side of (5.29) by the following identity:

$$\mathbf{f}[z]\mathbf{V}_1^1[z^{-1}] + \mathbf{f}[-z]\mathbf{V}_1^1[-z^{-1}] = 2\sum_{m=1}^{N/2}(\mathbf{f}\cdot\mathcal{T}_{2m-2}\mathbf{V}_1^1)z^{2m-2}. \tag{5.30}$$

Combining (5.30) with (5.28) yields

$$\mathbf{a}^1[z^2] = \frac{1}{2}\mathbf{f}[z]\mathbf{V}_1^1[z^{-1}] + \frac{1}{2}\mathbf{f}[-z]\mathbf{V}_1^1[-z^{-1}]. \tag{5.31}$$

Formula (5.31) is the desired relation between the polynomial $\mathbf{a}^1[z^2]$ and the z-transform of $\mathbf{f}$. Combining it with (5.27) yields

$$\mathbf{A}^1[z] = \frac{1}{2}\mathbf{f}[z]\mathbf{V}_1^1[z]\mathbf{V}_1^1[z^{-1}] + \frac{1}{2}\mathbf{f}[-z]\mathbf{V}_1^1[z]\mathbf{V}_1^1[-z^{-1}]. \tag{5.32}$$

In order to interpret Equation (5.32) correctly, we need to understand the effects of substituting z^{-1} and $-z$ into z-transforms. Substituting z^{-1} into the z-transform $\mathbf{f}[z]$ yields

$$\mathbf{f}[z^{-1}] = \sum_{m=1}^{N} f_m(z^{-1})^{m-1}.$$

We shall use the overbar notation, $\overline{\ }$, to denote the conjugation of complex numbers. Since z is on the unit-circle, we have $z^{-1} = \overline{z}$. And, because each value f_m of $\mathbf{f}$ is a real number, we have $f_m = \overline{f_m}$. Therefore

$$\begin{aligned}\mathbf{f}[z^{-1}] &= \sum_{m=1}^{N}\overline{f_m}\,\overline{z^{m-1}}\\ &= \overline{\mathbf{f}[z]}.\end{aligned}$$

Thus, for example, $\mathbf{V}_1^1[z]\mathbf{V}_1^1[z^{-1}] = \left|\mathbf{V}_1^1[z]\right|^2$.

To interpret the substitution of $-z$, we observe that $-z$ is the reflection of z through the origin of the complex plane. Consequently, the low frequency values that lie near $e^{i0} = 1$ are reflected into high frequency values that lie near $e^{\pm i\pi} = -1$, and vice versa. For example, in Figure 5.2, the spectrum $\left|\mathbf{V}_1^1[-z]\right|^2$ has a graph that is similar to the graph of the spectrum $\left|\mathbf{W}_1^1[z]\right|^2$ shown in Figure 5.2(b). The graph of $\left|\mathbf{V}_1^1[-z^{-1}]\right|^2$ is identical to the graph of $\left|\mathbf{V}_1^1[-z]\right|^2$, because the substitution of z^{-1} produces a complex conjugate which is then eliminated by the modulus-square operation. These considerations show that for the Coif12 scaling function, we have

$$\left|\mathbf{V}_1^1[z]\right|\,\left|\mathbf{V}^1[-z^{-1}]\right| \approx 0 \tag{5.33}$$

© 2008 by Taylor & Francis Group, LLC

except for two small intervals of values of z centered on $e^{\pm i\pi/2}$. The approximation in (5.33) is true for all of the orthogonal Daubechies scaling functions.

Based on (5.32) and (5.33) we have the following approximation:

$$\mathbf{A}^1[z] \approx \frac{1}{2}\left|\mathbf{V}_1^1[z]\right|^2 \mathbf{f}[z]. \tag{5.34}$$

Using the connection between DFTs and z-transforms, the approximation (5.20) follows from the approximation (5.34).

Similar calculations for the first-level detail signal $\mathbf{D}^1$ and fluctuation subsignal $\mathbf{d}^1$ yield

$$\mathbf{d}^1[z] = \frac{1}{2}\mathbf{f}[z]\mathbf{W}_1^1[z^{-1}] + \frac{1}{2}\mathbf{f}[-z]\mathbf{W}_1^1[-z^{-1}] \tag{5.35}$$

and hence

$$\mathbf{D}^1[z] = \frac{1}{2}\mathbf{f}[z]\mathbf{W}_1^1[z]\mathbf{W}_1^1[z^{-1}] + \frac{1}{2}\mathbf{f}[-z]\mathbf{W}_1^1[z]\mathbf{W}_1^1[-z^{-1}]. \tag{5.36}$$

And we also have the approximation

$$\mathbf{D}^1[z] \approx \frac{1}{2}\left|\mathbf{W}_1^1[z]\right|^2 \mathbf{f}[z], \tag{5.37}$$

which implies the approximation (5.21).

The other approximations, (5.22) through (5.25), can be proved in the same way as (5.20) and (5.21).

5.4 Correlation and feature detection

In this section we shall describe a standard method for detecting a short-lived feature within a more complicated signal. This method, known as correlation, is a fundamental part of Fourier analysis. Our discussion will set the stage for correlation and feature detection in 2D images, which we examine in the next section. We shall then also describe how wavelet analysis can be used to enhance the basic correlation method for feature detection.

Let's begin by examining feature detection for 1D signals. Feature detection is important in seismology, where there is a need to identify characteristic features that indicate, say, earthquake tremors within a long seismological signal. Or, in an electrocardiogram (ECG), it might be necessary to identify portions of the ECG that indicate an abnormal heartbeat.

At the top of Figure 5.5 we show a simulated ECG, which we shall refer to as Signal C. The feature that we wish to locate within Signal C is shown in the middle of Figure 5.5; this feature is meant to simulate a normal heartbeat. It is, of course, easy for us to visually locate the normal heartbeats within Signal C, but that is a far cry from an algorithm that a computer could use for automatic detection. Of course, once we locate the normal heartbeats, then the remaining heartbeats are abnormal.

© 2008 by Taylor & Francis Group, LLC

As noted above, the standard method used for feature detection is correlation. The *correlation* of a signal $\mathbf{f}$ with a signal $\mathbf{g}$, both having lengths of N values, will be denoted by $(\mathbf{f} : \mathbf{g})$. It is also a signal of length N, and its k^{th} value $(\mathbf{f} : \mathbf{g})_k$ is defined by

$$(\mathbf{f} : \mathbf{g})_k = f_1 g_k + f_2 g_{k+1} + \cdots + f_N g_{k+N-1}. \tag{5.38}$$

In order for the sum in (5.38) to make sense, the signal $\mathbf{g}$ needs to be periodically extended, i.e., we assume that $g_{k+N} = g_k$ for each k. When computing the correlation $(\mathbf{f} : \mathbf{g})$, the signal $\mathbf{f}$ is the feature that we wish to detect within the signal $\mathbf{g}$. Usually the signal $\mathbf{f}$ is similar to the normal heartbeat signal shown in the middle of Figure 5.5, in the sense that the values of $\mathbf{f}$ are 0 except near the central portion of the signal. This reduces the distortion that results from assuming that $\mathbf{g}$ is periodic. We will show later how correlations are related to Fourier analysis, but first we shall describe their use in feature detection.

The rationale behind using the correlation $(\mathbf{f} : \mathbf{g})$ to detect the location of $\mathbf{f}$ within $\mathbf{g}$ is the following. If a portion of $\mathbf{g}$ matches the form of the central portion of $\mathbf{f}$—where the significant, non-zero values are concentrated—then, for a certain value of k, the terms in (5.38) will all be squares. This produces a positive sum which is generally larger than the sums for the other values of $(\mathbf{f} : \mathbf{g})$. In order to normalize this largest value so that it equals 1, we shall divide the values of $(\mathbf{f} : \mathbf{g})$ by the energy of $\mathbf{f}$. That is, we denote the *normalized correlation* of $\mathbf{f}$ with $\mathbf{g}$ by $\langle \mathbf{f} : \mathbf{g} \rangle$, and the k^{th} value of $\langle \mathbf{f} : \mathbf{g} \rangle$ is

$$\langle \mathbf{f} : \mathbf{g} \rangle_k = \frac{f_1 g_k + f_2 g_{k+1} + \cdots + f_N g_{k+N-1}}{\mathcal{E}_{\mathbf{f}}}. \tag{5.39}$$

At the end of this section we shall discuss why, under the right conditions, the maximum value for $\langle \mathbf{f} : \mathbf{g} \rangle$ is approximately 1.

As an example of these ideas, we show at the bottom of Figure 5.5 the graph of those values of the normalized correlation $\langle \mathbf{f} : \mathbf{g} \rangle$ for the normal heartbeat and Signal C that exceed 0.90. Notice how *the maxima for this graph clearly locate the positions of the normal heartbeats within Signal C.* The value of these maxima are 1, thus providing the following simple criterion for locating a normal heartbeat: *if a normalized correlation value is significantly close to* 1*, say greater than* 0.90*, then a normal heartbeat is probably present at the location of this value.*

5.4.1 DFT method of computing correlations

There is a simple relationship between correlations and the DFTs of signals. We shall now give a brief sketch of this relationship. Since the normalized correlation $\langle \mathbf{f} : \mathbf{g} \rangle$ simply consists of dividing $(\mathbf{f} : \mathbf{g})$ by the energy of $\mathbf{f}$, we shall concentrate on the problem of computing $(\mathbf{f} : \mathbf{g})$.

The following formula, which we shall prove later, describes the relation-

© 2008 by Taylor & Francis Group, LLC

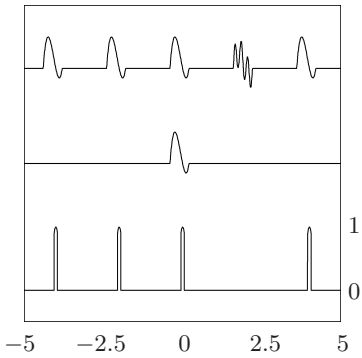


FIGURE 5.5
Top: Signal C. Middle: normal heartbeat. Bottom: values of normalized correlation that exceed 0.90**.**

ship between the DFT of $(\mathbf{f} : \mathbf{g})$ and the DFTs of $\mathbf{f}$ and $\mathbf{g}$:

$$(\mathbf{f} : \mathbf{g}) \overset{\mathcal{F}}{\longmapsto} \mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}} \tag{5.40}$$

where $\mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}}$ is the product signal with values $\left(\mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}}\right)_n = (\mathcal{F}\mathbf{f})_n\,\overline{(\mathcal{F}\mathbf{g})_n}$. This formula shows that the frequency content of $(\mathbf{f} : \mathbf{g})$ simply consists of the product of the DFT of $\mathbf{f}$ with the complex conjugate of the DFT of $\mathbf{g}$. Formula (5.40) also gives us the following three-step method for computing the correlation $(\mathbf{f} : \mathbf{g})$.

DFT calculation of correlation $(\mathbf{f} : \mathbf{g})$

Step 1. Compute DFTs of $\mathbf{f}$ and $\mathbf{g}$.

Step 2. Multiply the values of $\mathcal{F}\mathbf{f}$ and $\overline{\mathcal{F}\mathbf{g}}$ to produce $\mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}}$.

Step 3. Compute the DFT inverse of $\mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}}$ to produce $(\mathbf{f} : \mathbf{g})$.

Although this method may appear convoluted at first sight, it is actually much more efficient than a direct calculation of $(\mathbf{f} : \mathbf{g})$ based on Formula (5.38).

The reason that the DFT calculation of correlations is more efficient than Formula (5.38) is because an FFT algorithm is employed for performing the DFTs. Formula (5.38) requires N multiplications and $N - 1$ additions in order to calculate each of the N values of $(\mathbf{f} : \mathbf{g})$. That amounts to $2N^2 - N$ operations in total to compute $(\mathbf{f} : \mathbf{g})$. Computing the correlation $(\mathbf{f} : \mathbf{g})$ by this direct method is said to require $\mathcal{O}(N^2)$ operations; meaning that a bounded multiple of N^2 operations is needed. An FFT algorithm requires only $\mathcal{O}(N \log_2 N)$ operations, hence the DFT calculation of correlation also requires only $\mathcal{O}(N \log_2 N)$ operations. When N is large, say $N \geq 1024$, then $N \log_2 N$ is significantly smaller than N^2.

© 2008 by Taylor & Francis Group, LLC

The DFT calculation of correlation is even more efficient when 2D images are involved, as in the next section. For 2D images, say both N by N, the correlation defined in (5.48) in the next section requires $\mathcal{O}(N^4)$ operations if a direct calculation is performed. A DFT calculation, however, requires only $\mathcal{O}(N^2 \log_2 N)$ operations. The value of N need not be very large at all in order for $N^2 \log_2 N$ to be significantly smaller than N^4.

5.4.2 Proof of DFT effect on correlation *

Formula (5.40) can be proved easily using z-transforms. Making use of cyclic translation [see Section 5.2], we can rewrite (5.38) in the form

$$(\mathbf{f} : \mathbf{g})_k = \mathbf{f} \cdot \mathcal{T}_{k-1}\mathbf{g}. \tag{5.41}$$

Consequently, the z-transform of $(\mathbf{f} : \mathbf{g})$ is

$$\begin{aligned}(\mathbf{f} : \mathbf{g})[z] &= \sum_{k=1}^{N} (\mathbf{f} \cdot \mathcal{T}_{k-1}\mathbf{g})\, z^{k-1} \\ &= \mathbf{f}[z]\, \mathbf{g}[z^{-1}].\end{aligned} \tag{5.42}$$

The second equality in (5.42) holds for the same reason that Equality (5.29) holds. Since $\mathbf{g}$ is a real-valued signal, we have $\mathbf{g}[z^{-1}] = \overline{\mathbf{g}[z]}$; so (5.42) becomes

$$(\mathbf{f} : \mathbf{g})[z] = \mathbf{f}[z]\, \overline{\mathbf{g}[z]}. \tag{5.43}$$

From (5.43) and the relation between z-transforms and DFTs, we obtain (5.40).

5.4.3 Normalized correlations and feature detection *

In this subsection we shall briefly examine the mathematical justification for using normalized correlations to detect the presence of one signal within another, more complicated, signal. This discussion will make use of concepts from linear algebra—in particular, Cauchy's inequality for scalar products. Those readers who are not conversant with linear algebra should feel free to skip this subsection; we shall not be referring to it in the sequel.

Let $\mathbf{f}$ be a signal of positive energy, $\mathcal{E}_\mathbf{f} > 0$. We assume a positive energy in order to force $\mathbf{f} \neq (0, 0, \ldots, 0)$, because it is clearly pointless to try to detect the signal $(0, 0, \ldots, 0)$ within any signal. Suppose that the signal $\mathbf{g}$ contains the signal $\mathbf{f}$ in the sense that, for some integer m between 1 and N,

$$\mathbf{g} = \mathcal{T}_{m-1}\mathbf{f} + \mathbf{n} \tag{5.44}$$

where $\mathcal{T}_{m-1}\mathbf{f}$ is a cyclic translate of $\mathbf{f}$ and $\mathbf{n}$ is a *noise term.* By noise term we mean an undesired portion of the signal. The signal $\mathbf{n}$ is certainly undesired because we want to detect the presence of $\mathbf{f}$; however, the detection method

© 2008 by Taylor & Francis Group, LLC

based on normalized correlation works best when $\mathbf{n}$ is, in fact, random noise that is completely uncorrelated to $\mathbf{f}$. By completely uncorrelated to $\mathbf{f}$ we mean that

$$\langle \mathbf{f} : \mathbf{n} \rangle_j = \frac{\mathbf{f} \cdot \mathcal{T}_{j-1}\mathbf{n}}{\mathcal{E}_\mathbf{f}} \approx 0 \tag{5.45}$$

holds for each integer j.

Assuming that (5.45) holds, we now let $k = N - m + 2$ and note that $\mathcal{T}_{k-1} \circ \mathcal{T}_{m-1} = \mathcal{T}_N = \mathcal{T}_0$. Since $\mathcal{T}_0$ is the identity mapping, we then have

$$\begin{aligned} \langle \mathbf{f} : \mathbf{g} \rangle_k &= \frac{\mathbf{f} \cdot (\mathcal{T}_{k-1} \circ \mathcal{T}_{m-1}\mathbf{f})}{\mathcal{E}_\mathbf{f}} + \frac{\mathbf{f} \cdot \mathcal{T}_{k-1}\mathbf{n}}{\mathcal{E}_\mathbf{f}} \\ &= \frac{\mathbf{f} \cdot \mathbf{f}}{\mathcal{E}_\mathbf{f}} + \frac{\mathbf{f} \cdot \mathcal{T}_{k-1}\mathbf{n}}{\mathcal{E}_\mathbf{f}} . \end{aligned}$$

Because $\mathbf{f} \cdot \mathbf{f} = \mathcal{E}_\mathbf{f}$ and (5.45) holds, we then have

$$\langle \mathbf{f} : \mathbf{g} \rangle_k = 1 \tag{5.46}$$

except for a small error that (5.45) allows us to ignore.

On the other hand, if n is any of the integers between 1 and N, then letting $j = n + m - 2$ yields

$$\begin{aligned} \langle \mathbf{f} : \mathbf{g} \rangle_n &= \frac{\mathbf{f} \cdot \mathcal{T}_j\mathbf{f}}{\mathcal{E}_\mathbf{f}} + \frac{\mathbf{f} \cdot \mathcal{T}_{n-1}\mathbf{n}}{\mathcal{E}_\mathbf{f}} \\ &\approx \frac{\mathbf{f} \cdot \mathcal{T}_j\mathbf{f}}{\mathcal{E}_\mathbf{f}} . \end{aligned}$$

By the Cauchy inequality, we obtain

$$|\mathbf{f} \cdot \mathcal{T}_j\mathbf{f}| \le \sqrt{\mathcal{E}_\mathbf{f}} \sqrt{\mathcal{E}_{\mathcal{T}_j\mathbf{f}}} = \mathcal{E}_\mathbf{f}$$

and equality holds if and only if $\mathcal{T}_j\mathbf{f} = \mathbf{f}$ or $\mathcal{T}_j\mathbf{f} = -\mathbf{f}$. Therefore, except for a small error which (5.45) allows us to ignore, we have

$$-1 \le \langle \mathbf{f} : \mathbf{g} \rangle_n \le 1. \tag{5.47}$$

Equality holds on the right side of (5.47) only when $\mathbf{f} = \mathcal{T}_j\mathbf{f}$.

This discussion shows that we can detect the presence of cyclic translates of $\mathbf{f}$ within the signal $\mathbf{g}$ by the location of maximum value(s) of 1 among the positive values of $\langle \mathbf{f} : \mathbf{g} \rangle$. The method works best when those translates of $\mathbf{f}$ which are not equal to $\mathbf{f}$ produce scalar products that are much smaller than the energy of $\mathbf{f}$; this is the case, for instance, with the abnormal heartbeat considered above.

The discussion in this subsection can be extended to 2D images and will apply to the wavelet based method of object detection in images that we shall describe in the next section.

© 2008 by Taylor & Francis Group, LLC

5.5 Object detection in 2D images

In the previous section we described a basic method of 1D feature detection. In this section we shall describe how objects can be detected within 2D images. We shall discuss some examples related to character detection and the more difficult problem of locating small objects within complicated scenes.

Our first example shows how to identify the image of the character, P, shown in Gr 1 of Figure 5.6, within the sequence of three characters, PQW, shown in Gr 2 of Figure 5.6. One method for doing this is to compute a normalized correlation. For 2D images $\mathbf{f}$ and $\mathbf{g}$, a normalized correlation of $\mathbf{f}$ with $\mathbf{g}$ is defined as a correlation divided by the energy of $\mathbf{f}$, as we did for 1D signals in the previous section. To understand the 2D definition of correlation, we rewrite the 1D definition in (5.38) in the following form:

$$(\mathbf{f} : \mathbf{g})_k = \sum_{n=1}^{N} f_n g_{n+k-1}.$$

The 2D correlation $(\mathbf{f} : \mathbf{g})$ of two M by N images $\mathbf{f}$ and $\mathbf{g}$ is defined by

$$(\mathbf{f} : \mathbf{g})_{k,j} = \sum_{n=1}^{N} \sum_{m=1}^{M} f_{n,m} g_{n+k-1,m+j-1}. \tag{5.48}$$

Formula (5.48) requires a periodic extension of $\mathbf{g}$, i.e., $g_{n+N,m+M} = g_{n,m}$ is assumed to hold for all integers m and n. Based on (5.48), we define the normalized correlation $\langle \mathbf{f} : \mathbf{g} \rangle$ of two M by N images $\mathbf{f}$ and $\mathbf{g}$ by

$$\langle \mathbf{f} : \mathbf{g} \rangle_{k,j} = \frac{(\mathbf{f} : \mathbf{g})_{k,j}}{\mathcal{E}_{\mathbf{f}}}. \tag{5.49}$$

As with 1D signals, the identification of an object $\mathbf{f}$ within an image $\mathbf{g}$ is achieved by locating maximum values that are approximately 1 from among the positive values of $\langle \mathbf{f} : \mathbf{g} \rangle$. For example, if the image of P in Gr 1 of Figure 5.6 is the object $\mathbf{f}$ and the image of PQW in Gr 2 is the image $\mathbf{g}$, then we show the values that exceed 0.90 of $\langle \mathbf{f} : \mathbf{g} \rangle$ in Gr 4. The maximum value of 1 is visible as a tiny white dot that precisely locates the position of P within the image of PQW.

Our second example involves locating a small object within a complicated scene. This problem reveals the limitations of the correlation method; nevertheless, with some assistance from wavelet analysis, we shall outline an effective solution. In Gr 1 of Figure 5.7(a), we show a small image. We wish to identify the location, or the absence, of this object within the other three images in Gr 2 through Gr 4 of Figure 5.7(a). In Figure 5.7(b) we show the values that exceed 0.90 of the normalized correlations of the desired object with each of the images in Figure 5.7(a). It is clear from these Figures that

© 2008 by Taylor & Francis Group, LLC

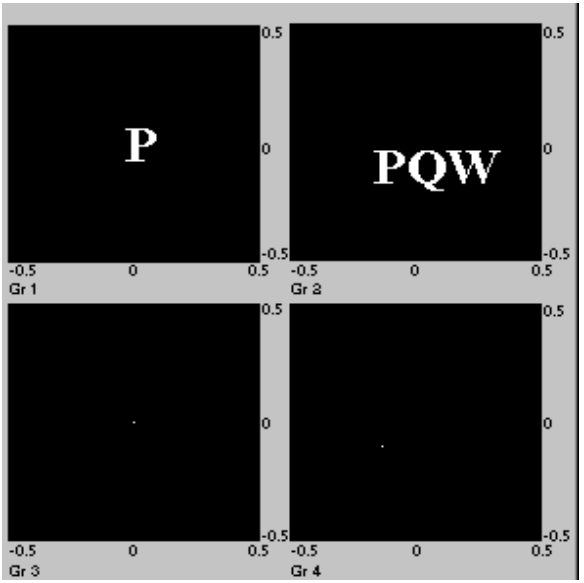


FIGURE 5.6
Gr 1: Image of P. Gr 2: Image of PQW. Gr 3: values of normalized correlation of Gr 1 with itself that exceed 0.90**. Gr 4: values of normalized correlation of Gr 1 with Gr 2 that exceed** 0.90**.**

the method fails; it successfully locates Gr 1 within Gr 1, but does not correctly locate Gr 1 within Gr 3. Furthermore, for Gr 2 and Gr 4, the maximum values of the normalized correlations are 1.32 and 1.43. The fact that these maximum values are significantly greater than 1 indicates a partial breakdown of the method, but the worst result is the fact that Gr 1 is not present at all within the images Gr 2 and Gr 4.

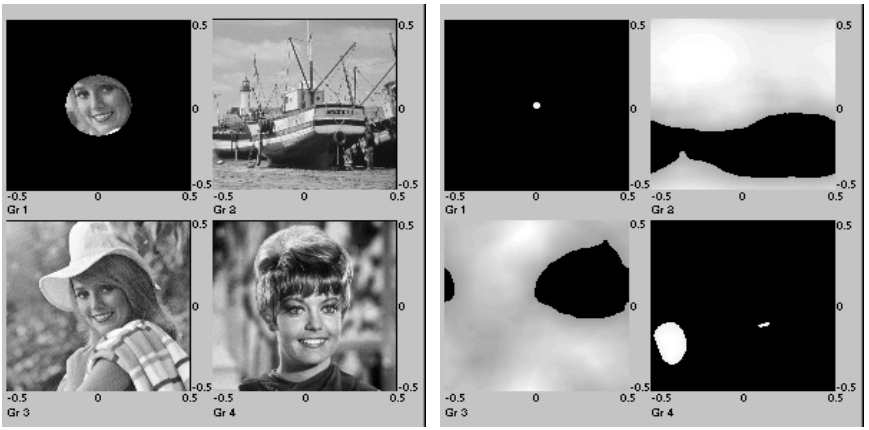
We shall now describe a wavelet based approach to identifying the location of this object. This approach rests on the fact that objects can often be identified via their edges; for example, there is some evidence that our visual system works on this basis. We saw at the end of the last chapter that the first-level detail signal $\mathbf{D}^1$ can provide us with an image consisting of edges. Based on these observations, we shall use the following three-step method for locating the object.

Edge Correlation Method of Object Location

Step 1. Compute a first-level detail image $\mathbf{D}^1$ for the object.

Step 2. Eliminate from this detail image any extraneous edge effects resulting from the outer edges in the object. This is not always necessary, but is needed with the object in Gr 1 in Figure 5.7(a). Removing its outer edge effects can be done by only retaining values that lie within a small enough radius.

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 5.7
(a) Four images, Gr 1 is the object that we wish to locate within the other three images. (b) Values that exceed 0.90 **of the normalized correlations of Gr 1 from (a) with each of the images in (a).**

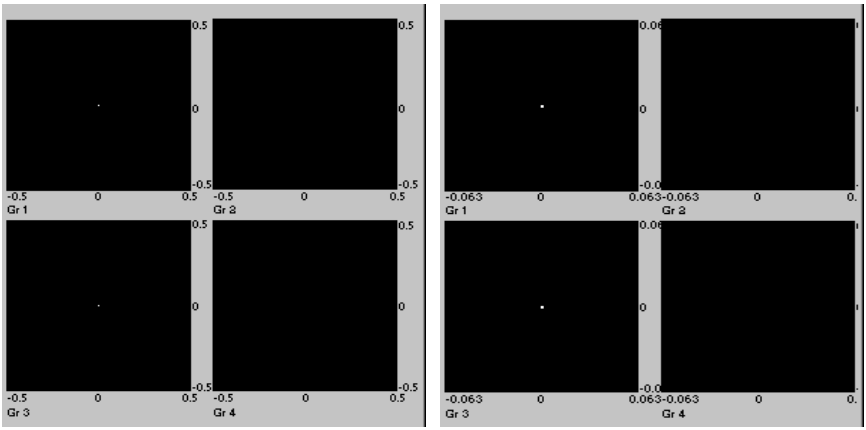
Step 3. Compute normalized correlations of the image from Step 2 with first-level detail signals for each of the other images. Determine if there are any values near 1 and where they are located.

In Figure 5.8(a) we show the results of this Edge Correlation Method, using Coif12 wavelets to produce the detail images. We have graphed values that exceed 0.90 of the normalized correlations. The location of the object within Gr 3 in Figure 5.7(a) is clearly indicated. The value of the normalized correlation at this point is 1. For Gr 2 and Gr 4 in Figure 5.8(a), there are no indications that the object is located within the corresponding images in Figure 5.7(a). In fact, the maximum values of the normalized correlations are 0.048 and 0.025, respectively, which are significantly less than 1.

We close this section with another example. In the last example, the object was in the same orientation within the image where we located it. Clearly this would not often be the case. For example, the object might be rotated through some angle, or it might be reflected about some line through the origin, or some combination of these last two operations. Let's refer to these operations as *symmetry operations* on the object. One solution to the problem of locating the object when it is in a different orientation is to perform the Edge Correlation Method on a finite set of objects obtained from performing a large number of symmetry operations on the initial object. The problem with this approach is that it is prohibitively time consuming.

We shall now outline a less time consuming solution to the problem just described. The gist of this solution is to perform the Edge Correlation Method on trend subimages for some specific level of wavelet transforms of the images. For example, in Figure 5.8(b), we show the values exceeding 0.90 of the nor-

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 5.8
Object detection: (a) Values exceeding 0.90 of the normalized correlations produced by the Edge Correlation Method applied to the images in Figure 5.7(a). (b) Similarly produced images resulting from the third trends of the images in Figure 5.7(a).

malized correlation images produced by the Edge Correlation Method applied to the third trend subimages $\mathbf{a}^3$ of each of the images in Figure 5.7(a). The location of the object within Gr 3 in Figure 5.7(a) is clearly indicated. The value of the normalized correlation at this point is 0.948, which is not 1 but very close to it. For Gr 2 and Gr 4 in Figure 5.8(b), there are no indications that the object is located within the corresponding images in Figure 5.7(a). In fact, the maximum values of the normalized correlations are 0.406 and 0.318, respectively, which are significantly less than 1.

The advantage of the method just described is that, by working with third level trend subimages, the sizes of the images are reduced by a factor of 64. This makes it feasible to perform a large number of symmetry operations on the object in order to search for rotated, or reflected, versions of it.

5.6 Creating scaling signals and wavelets *

In this section we outline the way in which scaling signals and wavelets are created from properties of their z-transforms. This material is more difficult than the other material in this chapter, and makes use of some earlier optional material; so those readers who wish to skip over it may certainly do so. No use will be made subsequently of this material.

The heart of wavelet theory is MRA, and so we begin by expressing the 1-level MRA equation $\mathbf{f} = \mathbf{A}^1 + \mathbf{D}^1$ in terms of z-transforms:

$$\mathbf{f}[z] = \mathbf{A}^1[z] + \mathbf{D}^1[z]. \tag{5.50}$$

© 2008 by Taylor & Francis Group, LLC

Using Formulas (5.32) and (5.36) in the right side of (5.50), we obtain

$$\mathbf{f}[z] = \mathbf{f}[z]\left\{\frac{1}{2}\mathbf{V}_1^1[z]\mathbf{V}_1^1[z^{-1}] + \frac{1}{2}\mathbf{W}_1^1[z]\mathbf{W}_1^1[z^{-1}]\right\} + \mathbf{f}[-z]\left\{\frac{1}{2}\mathbf{V}_1^1[z]\mathbf{V}_1^1[-z^{-1}] + \frac{1}{2}\mathbf{W}_1^1[z]\mathbf{W}_1^1[-z^{-1}]\right\}. \tag{5.51}$$

By comparing the two sides of (5.51) we see that the following two equations must hold:

$$\mathbf{V}_1^1[z]\mathbf{V}_1^1[z^{-1}] + \mathbf{W}_1^1[z]\mathbf{W}_1^1[z^{-1}] = 2 \tag{5.52a}$$

$$\mathbf{V}_1^1[z]\mathbf{V}_1^1[-z^{-1}] + \mathbf{W}_1^1[z]\mathbf{W}_1^1[-z^{-1}] = 0. \tag{5.52b}$$

In order to make (5.52b) hold, we define $\mathbf{W}_1^1[z]$ by

$$\mathbf{W}_1^1[z] = -z^{2k+1}\mathbf{V}_1^1[-z^{-1}] \tag{5.53}$$

where the exponent $2k+1$ is an odd integer that we shall specify later.

Before we go further, it is interesting to observe that (5.53) implies that

$$\left|\mathbf{W}_1^1[z]\right| = \left|\mathbf{V}_1^1[-z^{-1}]\right| \tag{5.54}$$

because z is on the unit-circle (so $|z| = 1$). Formula (5.54) implies the approximation (5.33) for the Coif12 case that we considered in Section 5.3, given the graphs of $\left|\mathbf{V}_1^1\right|^2$ and $\left|\mathbf{W}_1^1\right|^2$ shown in Figure 5.2(a).

We now return to our derivation of the z-transforms of scaling functions and wavelets. Combining (5.53) and (5.52a), and the identity

$$\mathbf{V}_1^1[z]\mathbf{V}_1^1[z^{-1}] = \left|\mathbf{V}_1^1[z]\right|^2,$$

we conclude that

$$\left|\mathbf{V}_1^1[z]\right|^2 + \left|\mathbf{V}_1^1[-z]\right|^2 = 2. \tag{5.55}$$

In order to satisfy (5.55) it is easier to work with the function $P(\theta)$ defined by

$$P(\theta) = \frac{1}{\sqrt{2}}\,\mathbf{V}_1^1[e^{i2\pi\theta}], \tag{5.56}$$

where θ is a real variable. Using this function $P(\theta)$, Equation (5.55) becomes

$$|P(\theta)|^2 + |P(\theta + 1/2)|^2 = 1. \tag{5.57}$$

We will now show how the Daub4 scaling numbers in Equation (3.3) can be obtained by solving Equation (5.57). We begin by observing that the following trigonometric identity

$$|\cos\pi\theta|^2 + |\cos\pi(\theta + 1/2)|^2 = 1 \tag{5.58}$$

© 2008 by Taylor & Francis Group, LLC

resembles the form of (5.57). In fact, if we were to set $P(\theta) = e^{i\pi\theta}\cos\pi\theta$, then we would be led to the two scaling numbers $\alpha_1 = \alpha_2 = 1/\sqrt{2}$ for the Haar scaling function $\mathbf{V}_1^1$. We leave the details to the reader; the reasoning involved is a simplified version of the argument that we shall now use to obtain the Daub4 scaling numbers.

The first step is to cube both sides of (5.58) obtaining

$$\begin{aligned} 1 &= \left(\cos^2\pi\theta + \sin^2\pi\theta\right)^3 \\ &= \cos^6\pi\theta + 3\cos^4\pi\theta\sin^2\pi\theta + 3\cos^2\pi\theta\sin^4\pi\theta + \sin^6\pi\theta. \end{aligned} \tag{5.59}$$

We now require that $|P(\theta)|^2$ satisfies

$$|P(\theta)|^2 = \cos^6\pi\theta + 3\cos^4\pi\theta\sin^2\pi\theta, \tag{5.60}$$

which are the first two terms on the right side of (5.59). The remaining two terms on the right side of (5.59) are equal to $|P(\theta+1/2)|^2$; so (5.57) holds.

Our final task is to obtain a function $P(\theta)$ which satisfies (5.60). In general, this is done via a result known as the *Riesz lemma.* This approach is described in the references on wavelets given at the end of the chapter. For the case of the Daub4 scaling functions, however, we can find $P(\theta)$ by a more direct, though still somewhat tricky, argument. We observe that

$$|P(\theta)|^2 = \cos^4\pi\theta\left[\cos^2\pi\theta + 3\sin^2\pi\theta\right]; \tag{5.61}$$

so we could set $P(\theta) = [\cos\pi\theta]^2\left[\cos\pi\theta - i\sqrt{3}\sin\pi\theta\right]$. For reasons that will be clear at the end, however, we instead define $P(\theta)$ by

$$P(\theta) = e^{i3\pi\theta}[\cos\pi\theta]^2\left[\cos\pi\theta - i\sqrt{3}\sin\pi\theta\right]. \tag{5.62}$$

Notice that, because $\left|e^{i3\pi\theta}\right|^2 = 1$, this formula for $P(\theta)$ implies that $P(\theta)$ satisfies (5.61). Since (5.61) holds, it follows that (5.60) does as well.

Our final task is to convert (5.62) into a form that allows us to read off the z-transform of $\mathbf{V}_1^1[z]$. To do this, we observe that

$$\begin{aligned} P(\theta) &= e^{i6\pi\theta}\left[e^{-i\pi\theta}\cos\pi\theta\right]^2\left[e^{-i\pi\theta}\cos\pi\theta - i\sqrt{3}e^{-i\pi\theta}\sin\pi\theta\right] \\ &= e^{i6\pi\theta}\frac{1}{4}\left[1+e^{-i2\pi\theta}\right]^2\left[\frac{1+e^{-i2\pi\theta}}{2} + \frac{\sqrt{3}}{2}\left(e^{-i2\pi\theta}-1\right)\right]. \end{aligned}$$

Multiplying out the last expression and simplifying yields the formula we need:

$$P(\theta) = \frac{1+\sqrt{3}}{8} + \frac{3+\sqrt{3}}{8}e^{i2\pi\theta} + \frac{3-\sqrt{3}}{8}e^{i4\pi\theta} + \frac{1-\sqrt{3}}{8}e^{i6\pi\theta}. \tag{5.63}$$

Since $\mathbf{V}_1^1[e^{i2\pi\theta}] = \sqrt{2}\,P(\theta)$, we obtain the z-transform $\mathbf{V}_1^1[z]$ from (5.63) by setting $z = e^{i2\pi\theta}$:

$$\mathbf{V}_1^1[z] = \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{3+\sqrt{3}}{4\sqrt{2}}z + \frac{3-\sqrt{3}}{4\sqrt{2}}z^2 + \frac{1-\sqrt{3}}{4\sqrt{2}}z^3. \tag{5.64}$$

© 2008 by Taylor & Francis Group, LLC

Formula (5.64) tells us that $\mathbf{V}_1^1$ is defined by

$$\mathbf{V}_1^1 = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, \ldots, 0)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are the Daub4 scaling numbers defined in Formula (3.3). We have thus shown how those scaling numbers are obtained via z-transform theory.

The definition of the Daub4 wavelet numbers now follows easily. If we set $k = 1$ in Formula (5.53), then $\mathbf{W}_1^1[z] = -z^3\mathbf{V}_1^1[-z^{-1}]$. This equation combined with (5.64) yields

$$\mathbf{W}_1^1[z] = \frac{1-\sqrt{3}}{4\sqrt{2}} + \frac{\sqrt{3}-3}{4\sqrt{2}}\, z + \frac{3+\sqrt{3}}{4\sqrt{2}}\, z^2 + \frac{-1-\sqrt{3}}{4\sqrt{2}}\, z^3. \tag{5.65}$$

Formula (5.65) implies that

$$\mathbf{W}_1^1 = (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots, 0)$$

where $\beta_1, \beta_2, \beta_3, \beta_4$ are the Daub4 wavelet numbers defined in Formula (3.8).

We end this section by noting that other properties of wavelets can be obtained by requiring that the function $P(\theta)$ satisfies certain identities. For instance, the condition that the Daub4 scaling numbers satisfy

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = \sqrt{2}$$

is equivalent to the requirement that

$$P(0) = 1. \tag{5.66}$$

Notice that the function $P(\theta)$ defined above does satisfy this requirement. Furthermore, the conditions on the Daub4 wavelet numbers, stated in Equations (3.11) and (3.12), can be easily seen to be equivalent to the two equations

$$\mathbf{W}_1^1[1] = 0, \quad \frac{d}{dz}\mathbf{W}_1^1[1] = 0. \tag{5.67}$$

Tracing back through the definitions, it is not hard to show that these last two equations are equivalent to

$$P(1/2) = 0, \quad \frac{dP}{d\theta}(1/2) = 0. \tag{5.68}$$

These equations are, indeed, satisfied by the function $P(\theta)$ defined above.

Equations (5.66) and (5.68) are important because they show how crucial identities involving the scaling numbers and wavelets can be expressed in terms of values of $P(\theta)$ and its derivatives at $\theta = 0$ and $\theta = 1/2$. For example, the function $P(\theta)$ for the Coif6 case is required to satisfy

$$P(0) = 1, \quad \frac{dP}{d\theta}(0) = 0, \quad \frac{d^2P}{d\theta^2}(0) = 0 \tag{5.69}$$

© 2008 by Taylor & Francis Group, LLC

and

$$P(1/2) = 0, \quad \frac{dP}{d\theta}(1/2) = 0. \tag{5.70}$$

The equations in (5.69) correspond to Equations (3.33a) through (3.33c), while the equations in (5.70) correspond to Equations (3.32a) and (3.32b).

5.7 Gabor transforms and spectrograms

Our treatment of frequency analysis would not be complete without discussing the most up-to-date methods involving *dynamic* spectral analysis, analyzing how the spectrum of a signal evolves in time. This analysis is carried out with *Gabor transforms,* which are close relatives of wavelet transforms. Gabor transforms are widely employed in the fields of speech processing, communications theory, audio design, and music (where they are also referred to as either *Short Time Fourier Transforms* or *sonograms.*) In this section we provide the basic mathematical description of Gabor transforms, while in subsequent sections we illustrate their application to musical analysis, musical synthesis, and denoising.

Gabor transforms are particularly useful in analyzing musical signals. To motivate their definition we cannot do better than to quote from one of the founders of the theory, Jean Ville:

> If we consider a passage [of music] containing several measures (which is the least that is needed) and if a note, *la* for example, appears once in the passage, harmonic [Fourier] analysis will give us the corresponding frequency with a certain amplitude and a certain phase, without localizing the *la* in time. But it is obvious that there are moments during the passage when one does not hear the *la.* The [Fourier] representation is nevertheless mathematically correct because the phases of the notes near the *la* are arranged so as to destroy this note through interference when it is not heard and to reinforce it, also through interference, when it is heard; but if there is in this idea a cleverness that speaks well for mathematical analysis, one must not ignore the fact that it is also a distortion of reality; indeed when the *la* is not heard, the true reason is that *la* is not emitted. Thus it is desirable to look for a mixed definition of a signal of the sort advocated by Gabor: at each instance, a certain number of frequencies are present, giving volume and timbre to the sound as it is heard; each frequency is associated with a certain partition of time that defines the intervals during which the corresponding note is emitted.[3]

As an example of what Ville means, consider the signal defined by taking 8192 uniform samples of the following function

$$g(t) = e^{-400(t-0.2)^2} \sin 1024\pi t + e^{-400(t-0.5)^2} \cos 2048\pi t + e^{-400(t-0.7)^2} (\sin 512\pi t - \cos 3072\pi t)$$

[3]From [2], page 63.

© 2008 by Taylor & Francis Group, LLC

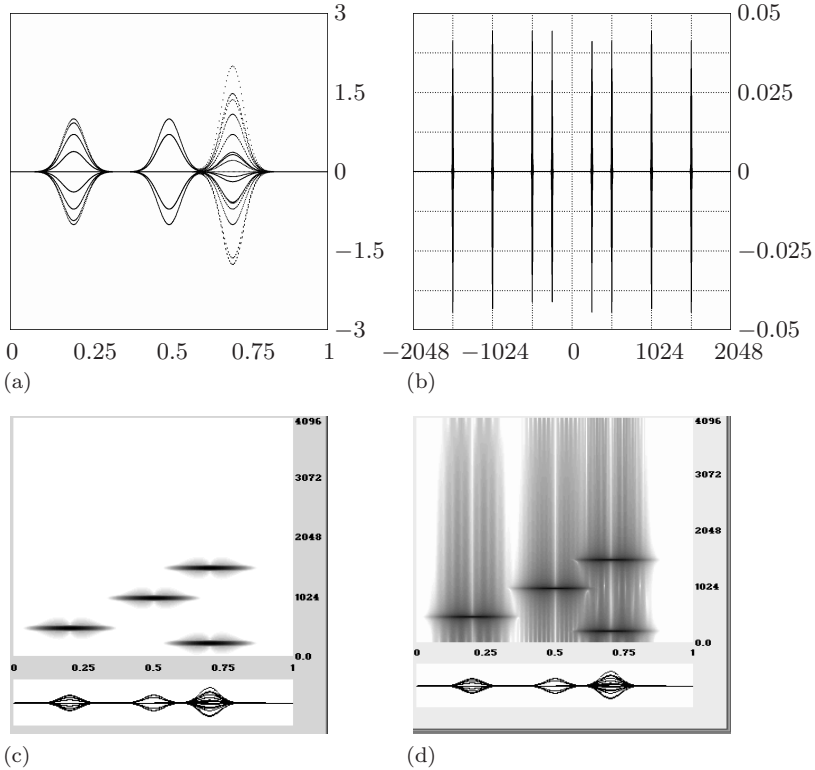


FIGURE 5.9
(a) Test signal. (b) DFT of test signal. (c) Blackman windowed spectrogram of test signal; time is in seconds along the horizontal, frequency is in Hz along the vertical. (d) Boxcar windowed spectrogram of test signal. The Blackman window produces a good spectrogram, while the Boxcar window produces a poor spectrogram.

over the time interval $[0, 1]$. When played over a computer sound system, this signal sounds like three flute-like notes. We have graphed this signal in Figure 5.9(a). The exponentials produce the "bump" shaped portions of this graph as they decrease rapidly to 0 when t moves away from 0.2, 0.5, or 0.7. The sines and cosines that multiply each exponential produce oscillations of frequency 512, 1024, 256, and 1536. In Figure 5.9(b) we show the DFT of this signal. Notice that these frequencies are identified by the positions of the spikes in the DFT. As alluded to by Ville, however, it is quite difficult to infer any information about the time locations of the emitted notes from the DFT. In contrast, consider Figure 5.9(c). In that figure, we have graphed a spectrogram (the magnitudes-squared of a Gabor transform) as advocated by Ville. The dark horizontal bars lie above the notes in the signal and at heights corresponding exactly to the frequencies of those notes. This spectrogram provides a dynamic time-frequency portrait of this musical signal.

© 2008 by Taylor & Francis Group, LLC

We now give a precise definition of Gabor transforms and spectrograms. Suppose that our signal $\mathbf{f}$ consists of discrete samples of an analog signal $g(t)$. That is $\{f_k\} = \{g(t_k)\}_{k=0}^{N}$, for uniformly spaced values $t_k = k\Delta t$ in a finite interval $[0, T]$. A *Gabor transform* of f, with window function w, is defined as follows. First, multiply $\{g(t_k)\}$ by a sequence of shifted window functions $\{w(t_k - \tau_m)\}_{m=0}^{M}$, producing time localized subsignals, $\{g(t_k)w(t_k - \tau_m)\}_{m=0}^{M}$. Uniformly spaced values, $\{\tau_m = t_{jm}\}_{m=0}^{M}$, are used for shifts ($j$ is a positive integer greater than 1). The supports for the windows $\{w(t_k - \tau_m)\}_{m=0}^{M}$ are all of finite extent and overlap each other. See Figure 5.10. The value of M is determined by the minimum number of windows needed to cover $[0, T]$, as illustrated in Figure 5.10(b). Note that because the values t_k belong to a finite interval $[0, T]$, we always extend our signal values beyond the interval's endpoints by appending zeroes, hence the full supports of all windows are included.

Two common window functions are Hanning and Blackman windows. The *Hanning window* w is defined by

$$w(t) = \begin{cases} 0.5 + 0.5\cos(2\pi t/\lambda) & \text{for } |t| \le \lambda/2 \\ 0 & \text{for } |t| > \lambda/2 \end{cases}$$

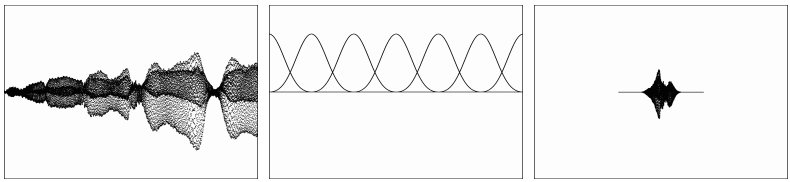
where λ is a parameter that gives the length of the support of the Hanning window. See Figure 5.11(a). The *Blackman window* w is defined by

$$w(t) = \begin{cases} 0.42 + 0.5\cos(2\pi t/\lambda) + 0.08\cos(4\pi t/\lambda) & \text{for } |t| \le \lambda/2 \\ 0 & \text{for } |t| > \lambda/2 \end{cases}$$

where λ again equals the length of the support of the window. See Figure 5.11(b).

Now, for the second step. Because each subsignal $\{g(t_k)w(t_k - \tau_m)\}$ has a finite support, we can apply DFTs to them *over the length of their supports.* See Figure 5.10(c). Denoting these DFTs by $\mathcal{F}$, we then have the Gabor transform of $\{g(t_k)\}$:

$$\{\mathcal{F}\{g(t_k)w(t_k - \tau_m)\}\}_{m=0}^{M}. \tag{5.71}$$



(a) (b) (c)

FIGURE 5.10
(a) Signal. (b) Succession of shifted window functions. (c) Signal multiplied by middle window in (b); a DFT can now be applied to this signal.

© 2008 by Taylor & Francis Group, LLC

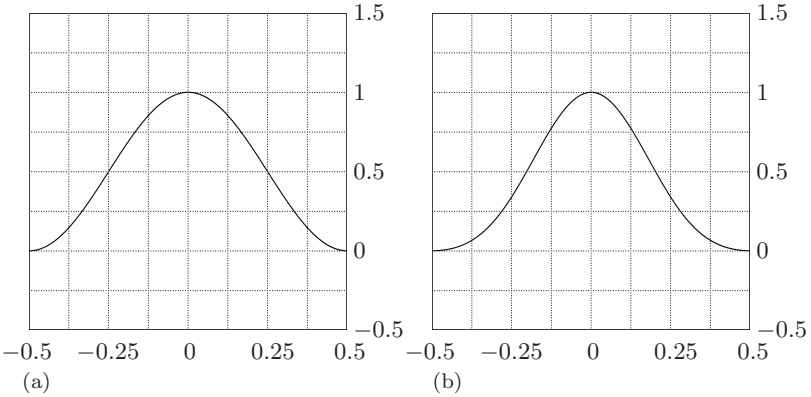


FIGURE 5.11
(a) Hanning window, $\lambda = 1$. (b) Blackman window, $\lambda = 1$.

This succession of shifted window functions $\{w(t_k - \tau_m)\}$ provides the *partitioning of time* referred to by Ville, and the DFTs provide the frequency analysis of the signal relative to this partition.

When displaying a Gabor transform, it is standard practice to display a plot of its magnitude-squared values, with time along the horizontal axis, frequency along the vertical axis, and darker pixels representing higher square-magnitudes. We shall refer to such a plot as a *spectrogram.* The spectrogram in Figure 5.9(c) was obtained using a Blackman window. Spectrograms for other Blackman windowed Gabor transforms are given in the next section in our analysis of musical signals.

The reader may wonder about using a window that damps down to 0 at the ends, like the Blackman or Hanning window. Why not just use a rectangular window? That is, the window

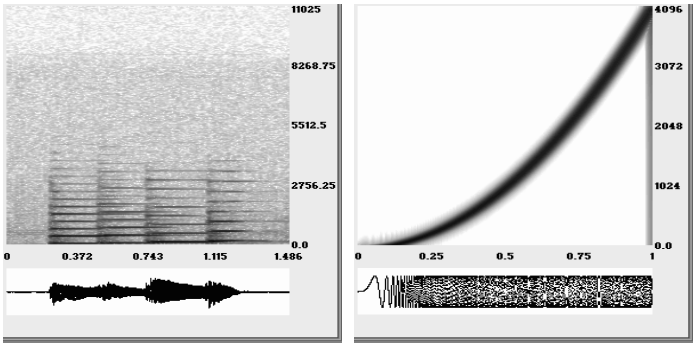
$$w(t) = \begin{cases} 1 & \text{for } |t| \le \lambda/2 \\ 0 & \text{for } |t| > \lambda/2 \end{cases}$$

which is called a *Boxcar* window (or rectangular window). The problem with using a Boxcar window is that its jump discontinuities at $\pm\lambda/2$ cause unacceptable distortions when multiplying the signal. For example, in Figure 5.9(d) we show a Boxcar windowed spectrogram of the test signal considered above. It is easy to see that the Blackman windowed spectrogram in Figure 5.9(c) provides a much better time-frequency description of the signal. The Boxcar windowed spectrogram shows far too much distortion as seen, for example, in the long vertical swatches of gray pixels arising from the artificial discontinuities introduced by multiplying by the Boxcar window.

5.8 Musical analysis

Having introduced the basic definitions of Gabor transform theory in the

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 5.12
(a) Spectrogram of several piano notes. (b) Spectrogram of artificial chirp.

previous section, we now turn to a beautiful application of it to the field of musical theory. We have chosen to analyze music in some detail because it lies at the intersection of art and science, and we hope to illustrate how these two domains can enhance rather than oppose each other.

A very succinct summary of the role of Gabor transforms in musical analysis is given by Monika Dörfler in her thesis:

> Diagrams resulting from time-frequency analysis ... can even be interpreted as a generalized musical notation.[4]

As an elaboration of this idea, we offer the following principle:

Multiresolution Principle. *Music is a patterning of sound in the time-frequency plane. Analyze music by looking for repetition of patterns of time-frequency structures over multiple time scales, and multiple resolution levels in the time-frequency plane.*

A simple illustration of this Multiresolution Principle is the structure of notes played on a piano. In Figure 5.12(a) we show a Blackman-windowed spectrogram of a succession of four piano notes. The division of the spectrogram into four sections (the four notes) along the time axis is evident in this figure. For each note, there are horizontal bars in the spectrogram that occur at integral multiples of a base frequency. For instance, in Figure 5.13(a) we show a graph of the magnitudes of a vertical slice of the Gabor transform values at a particular time, $t = 0.6$, corresponding to a point lying near the middle of the second note. This graph is the spectrum of the second piano note at $t = 0.6$. Notice that the main spikes in this spectrum occur at integral multiples of the base frequency 345 Hz. This base frequency is called the *fundamental* and the integral multiples of it are the *overtones.* The *first overtone* equals 2×345, the *second overtone* equals 3×345, and so on. Similarly, in Figure 5.13(b) we

[4]From [3], page xii.

© 2008 by Taylor & Francis Group, LLC

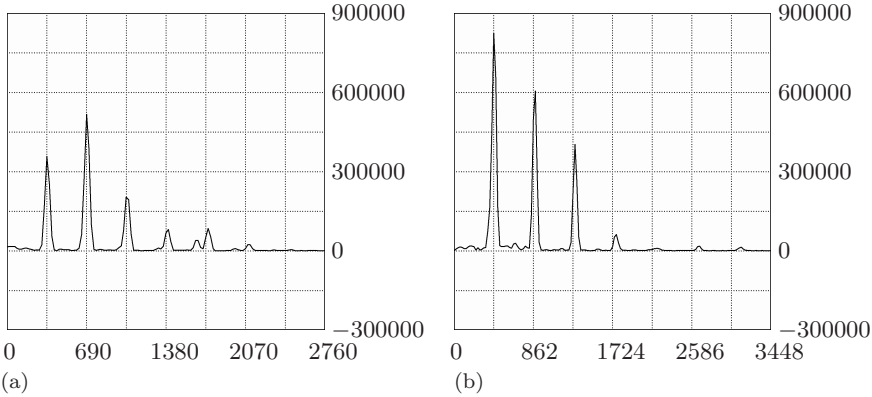


FIGURE 5.13
(a) Spectrum of piano note, $t = 0.6$. The fundamental is 345 Hz. (b) Spectrum of piano note, $t = 1.115$. The fundamental is 431 Hz.

show the magnitudes of the Gabor transform at $t = 1.115$, which provide the spectrum for the fourth piano note at this new time value. For this spectrum, the fundamental is 431 Hz, a higher pitch than the first spectrum. This patterning of structures in the time-frequency plane, horizontal bars at integral multiples of fundamental frequencies, is a basic feature of the tones produced by musical instruments, and has been given as an explanation of the pleasing effect of the notes produced by these instruments.[5]

For a second example, we look at a non-musical signal. In Figure 5.12(b) we show a spectrogram for an artificial signal known as a *chirp.* This chirp signal was generated from 8192 uniformly spaced samples $\{g(t_k)\}$ of the function

$$g(t) = \sin[8192(\pi/3)t^3]$$

over the interval $[0, 1]$. Chirp signals are used in Doppler radar tracking. Somewhat similar sounds are used by bats for echo navigation. When played on a computer sound system this chirp signal produces a kind of "alarm" sound of sharply rising pitch that most people would not classify as musical. Notice that there is very little if any "repetition of time-frequency structures at multiple time scales" as called for by our Multiresolution Principle.

5.8.1 Analysis of Stravinsky's *Firebird Suite*

We now turn to a more complex illustration of our Multiresolution Principle: an analysis of the famous ending passage of Stravinsky's *Firebird Suite.* Our

[5]The similarity of these horizontal bands of overtones to similar banding in the time-frequency structure of speech (see Figure 6.6 and its discussion in section 6.4) is striking. Musical instruments have long been regarded as amplifiers and extenders of human voice, and this may be a partial explanation for music's ability to emotionally affect us.

© 2008 by Taylor & Francis Group, LLC

discussion is based on one of the case studies in the article, "Music: a time-frequency approach" [4].

In Figure 5.14 we show a spectrogram of a clip from the ending passage of the *Firebird Suite,* with labelling of several important time-frequency structures. The left half of the spectrogram is relatively simple. It corresponds to a horn playing notes of the main theme for the passage with a faint string background of constant tone. We have marked the parts of the spectrogram corresponding to these musical structures. The constant tonal background is represented by long line segments marked $\mathbf{B}$ and $\mathbf{B}_1$ which represent the fundamental and first overtone of the constant tonal background. The horn notes are represented by the structure $\mathbf{T}$ for the fundamentals, and structure $\mathbf{T}_1$ above it for the first overtones. Higher overtones of the horn notes are also visible above these two structures.

The right half of the spectrogram in Figure 5.14 is much more complex. It is introduced by a harp glissando, marked as structure $\mathbf{G}$. That structure consists of a series of closely spaced horizontal bars whose left endpoints trace a steeply rising curve. The notes of this glissando are repeated over a longer time scale—marked as structure $\widetilde{\mathbf{G}}$. This structure $\widetilde{\mathbf{G}}$ is a *prolongation* of the glissando $\mathbf{G}$; its relation to $\mathbf{G}$ is emphasized by faint harp notes included among the notes of $\widetilde{\mathbf{G}}$. Here we see a prime example of repetition at multiple time scales. Another example of such repetition is that, while the prolongation is played, the main theme is repeated by the string section of the orchestra at a higher pitch than the horn. This latter repetition is marked by $\mathbf{T}_1$, and above it $\mathbf{T}_2$, which comprise the fundamentals and first overtones of the string notes. Following those structures there is again a repetition of time-frequency structure: the second glissando $\mathbf{G}$ which is played by the string section of the orchestra. This second glissando introduces a repetition of the main theme

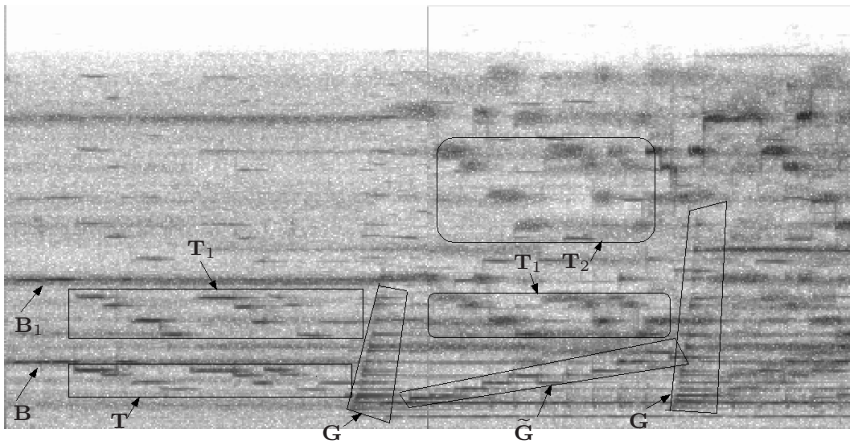
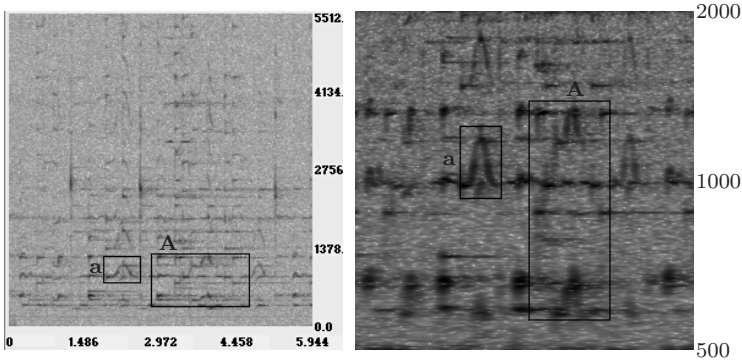


FIGURE 5.14
Spectrogram of passage from *Firebird suite* with important time-frequency structures marked (see text for explanation).

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 5.15
Time-frequency analysis of a classical Chinese folk melody. (a) Spectrogram. (b) Zooming in on two octaves of the frequency range of (a). The marked time-frequency structures are explained in the text.

at a new higher pitch played by a flute, along with orchestral accompaniment as a second prolongation of **G**, a repetition (with some embellishment) of the prolongation $\widetilde{\mathbf{G}}$. We have not marked these latter structures in the spectrogram, but they should be clearly visible to the reader.[6]

We encourage the reader to play the recording of the ending of the *Firebird Suite* which we have just analyzed. It is available as `firebird_clip2.wav` at the FAWAV website. An excellent way to play it is with the freeware AUDACITY, which can be downloaded from the website in [5]. AUDACITY allows you to trace out the spectrogram *as the music is playing.* Doing that will confirm the details of the analysis above, as well as providing a new appreciation of the beauty of the piece.

5.8.2 Analysis of a Chinese folk song

To illustrate the range of applicability of our Multiresolution Principle, we provide a brief analysis of a passage of classical Chinese folk music (available as `Chinese_Folk_Music.wav` at the book's website). In Figure 5.15(a), we show its spectrogram over a frequency range of 0 to 5512 Hz, with two important time-frequency structures labeled **a** and **A**. Structure **A** is an enlarged version of structure **a** created by repeating smaller scale versions of **a**. This is a perfect example of repetition of time-frequency structures at multiple time-scales. In (b) we show a zooming in on a 2-octave range of frequencies, from 500 to 2000 Hz, of the spectrogram in (a)—here we can see more clearly the repetition of patterns.[7] These time-frequency structures closely resemble the curved time-

[6] One final observation on this piece: notice that the repetition of the main theme, $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, follows a rising arc in the time-frequency plane that repeats, over a longer time-scale, the rising arc of $\widetilde{\mathbf{G}}$. This is a perfect illustration of the Multiresolution Principle (and also of the hierarchical structure of music described in the quote from Pinker on p. 200).

[7] See Section 5.12 for more details on how we created Figure 5.15(b).

© 2008 by Taylor & Francis Group, LLC

frequency bands occurring in speech, especially lyrics in song. An illustration from the lyrics of the song *Buenos Aires* is discussed on page 236, see especially Figure 6.8.

Of course, our Multiresolution Principle can only go so far in analyzing music. It represents an essential core of more elaborate principles formulated by Ray Jackendoff and Fred Lerdahl in their classic *Generative Theory of Tonal Music* [6]. Their theory is succinctly described by Steven Pinker as follows:

> Jackendoff and Lerdahl show how melodies are formed by sequences of pitches that are organized in three different ways, all at the same time... The first representation is a grouping structure. The listener feels that groups of notes hang together in motifs, which in turn are grouped into lines or sections, which are grouped into stanzas, movements, and pieces. This hierarchical tree is similar to a phrase structure of a sentence, and when the music has lyrics the two partly line up... The second representation is a metrical structure, the repeating sequence of strong and weak beats that we count off as "ONE-two-THREE-four." The overall pattern is summed up in musical notation as the time signature... The third representation is a reductional structure. It dissects the melody into essential parts and ornaments. The ornaments are stripped off and the essential parts further dissected into even more essential parts and ornaments on them... we sense it when we recognize variations of a piece in classical music or jazz. The skeleton of the melody is conserved while the ornaments differ from variation to variation.[8]

The similarity between the first and third representations described by Pinker and wavelet MRA is striking. It is why we referred to Gabor transforms as close relatives of wavelet transforms. Gabor transforms and spectrograms are used, together with the three representations described by Pinker, to provide a deeper analysis of musical structure in reference [4]. That article also extends Jackendoff and Lerdahl's theory to rhythmic aspects of music as well (we shall discuss this point further in Section 6.5).

We conclude our treatment of musical analysis with a couple of non-human examples of music from the realm of bird song. Bird song has long been recognized for its musical qualities. In Figure 5.16 we show spectrograms of the songs of an oriole and an osprey. The song of the osprey illustrates our Multiresolution Principle very well. In fact, there seems to be a repetition of two basic structures, labeled **A** and **B** (along with overtones) in Figure 5.16(a). The osprey's song can be transcribed as

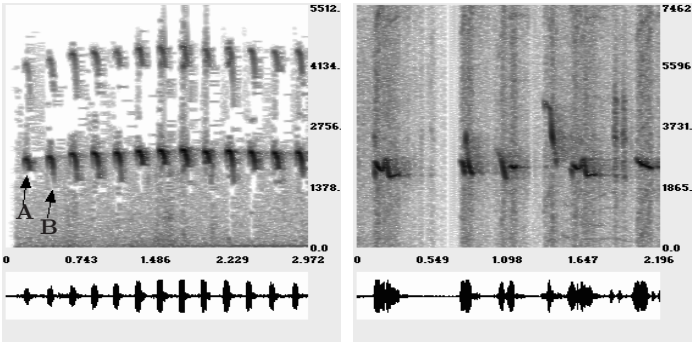
A B B B B A A A A B A A A

which reminds us of rhyming patterns in poetry.

The oriole song's spectrogram in Figure 5.16(b) is more richly structured. In contemplating it, we recall Dr. Dörfler's remark about *generalized musical*

[8] From [7], pages 532–533.

© 2008 by Taylor & Francis Group, LLC



(a) Osprey (b) Oriole

FIGURE 5.16
Spectrograms of bird songs.

notation. We leave it as an exercise for the reader to break up this spectrogram into component chirps using our Multiresolution Principle.

Both of these bird songs are available at the book's website [8]. The recording of the oriole's song has intermittent noise that interferes with the clarity of the individual chirps in the song. We will show how to denoise this recording in Section 5.10. But first we need to describe how Gabor transforms are inverted.

5.9 Inverting Gabor transforms

In this section we describe how Gabor transforms can be inverted, and how this inversion can be used for musical synthesis. In the next section we shall use inversion of Gabor transforms for denoising audio.

In order to perform inversion, we shall assume that there are two positive constants A and B such that the window function w satisfies

$$A \leq \sum_{m=0}^{M} w^2(t_k - \tau_m) \leq B \tag{5.72}$$

for all time values t_k. These two inequalities are called the *frame conditions* for the window function. For all of the windows employed by FAWAV these frame conditions do hold. The second inequality implies that the process of computing a Gabor transform with the window function w is numerically stable (does not cause computing overflow) whenever B is not too large. The first inequality is needed to ensure that inversion is also numerically stable.

We now show how to invert a Gabor transform

$$\{\mathcal{F}\{g(t_k)w(t_k - \tau_m)\}\}_{m=0}^{M}$$

of a signal $\{g(t_k)\}$. By applying DFT-inverses we obtain a set of subsignals

$$\{\{g(t_k)w(t_k - \tau_m)\}\}_{m=0}^{M}.$$

© 2008 by Taylor & Francis Group, LLC

For each m, we then multiply the m^{th} subsignal by $\{w(t_k - \tau_m)\}$ and sum over m, obtaining

$$\{\sum_{m=0}^{M} g(t_k)w^2(t_k - \tau_m)\} = \{g(t_k) \sum_{m=0}^{M} w^2(t_k - \tau_m)\}.$$

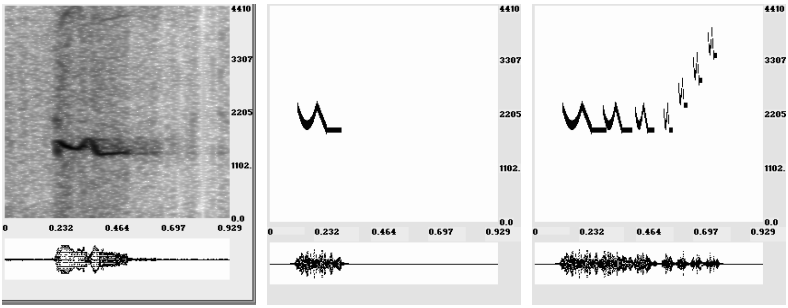
Multiplying the right side of this last equation by the values

$$\left[\sum_{m=0}^{M} w^2(t_k - \tau_m)\right]^{-1},$$

which by the frame condition (5.72) are no larger than A^{-1}, we obtain our original signal values $\{g(t_k)\}$. Thus, we have inverted our Gabor transform.

An interesting application of inversion is to the synthesis of musical passages. We now provide one illustration of such a synthesis. In Figure 5.17(a) we show a spectrogram of an oriole chirp, extracted from the beginning of the oriole's song that we discussed in the previous section. We used this spectrogram as our model for synthesizing an artificial bird chirp. That synthesis was begun by plotting the function described in the formula file `synthetic_oriole_whistle.uf2` using the plotting procedure of the spectrogram tool in FAWAV. This produced a parabolic segment, connected to a steeply sloping line segment, connected in turn to a horizontal line segment. See Figure 5.17(b). We then applied Gabor transform inversion to produce an audio signal, which we saved as `synthetic_bird_chirp.wav` at [8]. When played, it does indeed sound like a bird chirp.

As an isolated sound there is not much musicality to a single bird chirp. That follows from our Multiresolution Principle for musical structure in the time-frequency plane. There is simply not enough of a pattern in the time-frequency structure of this artificial bird chirp for us to perceive it as music. To create a more musical signal, an artificial bird song, we used this artificial bird chirp as a template for creating a pattern of repeating structures in the time-frequency plane. See Fig. 5.17(c). To produce this graph



(a) Oriole chirp (b) Synthetic bird chirp (c) Synthetic bird song

FIGURE 5.17
Spectrograms illustrating musical synthesis.

© 2008 by Taylor & Francis Group, LLC

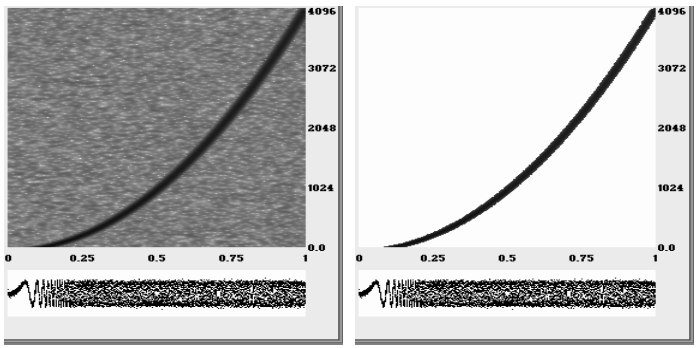
we repeated the time-frequency structure in Fig. 5.17(b) twice more at successively shrunken time-scales, followed by four more repetitions at a much shorter time-scale and translated upward in frequency. Our Multiresolution Principle predicts that a signal with such a time-frequency graph will sound musical. Indeed, the signal obtained by Gabor transform inversion (saved as `a_synthetic_bird_song.wav` at [8]) does sound like a bird song.

Of course, synthesizing bird song is only the beginning, although we would like to emphasize that bird song has inspired composers in the past. In the Notes and references section, we provide more details on the relation between musical composition and bird song, including pointers to some lovely avian-inspired music on the Internet. We also supply a reference to software for synthesizing beautiful electronic music from Gabor transforms.

5.10 Gabor transforms and denoising

An important application of inversion of Gabor transforms is noise removal. The basic idea is similar to wavelet denoising: apply thresholding to remove noisy transform values while preserving signal-dominated transform values. In this section we provide a brief introduction to this vast field. We discuss a couple of illustrative examples, and then turn to some examples of denoising real audio.

The essential idea behind threshold denoising of Gabor transforms can be illustrated with spectrograms. For example, in Figure 5.18(a) we show a spectrogram for a Gabor transform of the artificial chirp signal after random Gaussian noise has been added. Compare it with the one for the uncontaminated signal shown in Figure 5.12(b). As you can see, the noise appears as a background of gray pixels and there is a parabolic arc of signal-dominated transform values that clearly stands out from this background. In Figure 5.18(b) we show the spectrogram for the Gabor transform obtained by the thresholding process referenced below. Applying our inverse process to this latter



(a) (b)

FIGURE 5.18
(a) Noisy chirp spectrogram. (b) Thresholded spectrogram [c.f. Figure 5.12(b)].

© 2008 by Taylor & Francis Group, LLC

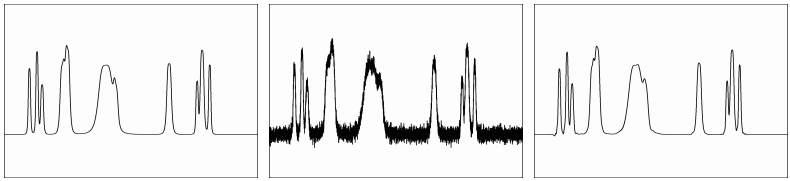


FIGURE 5.19
(a) Bumps signal. (b) Bumps signal with added noise, MSE = 1.00. (c) Thresholded Gabor denoising, MSE = 0.02.

transform, we obtained a good approximation of the uncontaminated artificial chirp. In fact the MSE is reduced from 1.00 for the noisy signal to 0.05 for the denoised signal.

Another example of the effectiveness of Gabor transform thresholding is shown in Figure 5.19. In Figure 5.19(a) we show a test signal called *Bumps.* This is a standard test signal used in signal processing. It is designed to simulate a signal obtained from a Nuclear Magnetic Resonance measurement. Figure 5.19(b) shows a signal generated by adding Gaussian noise of standard deviation 1 to the Bumps signal. Employing the thresholding process referenced below, we obtained the denoised signal shown in Figure 5.19(c). It is an excellent approximation of the original Bumps signal. In fact, the MSE is reduced from 1.00 for the noisy signal to 0.02 for the denoised signal.

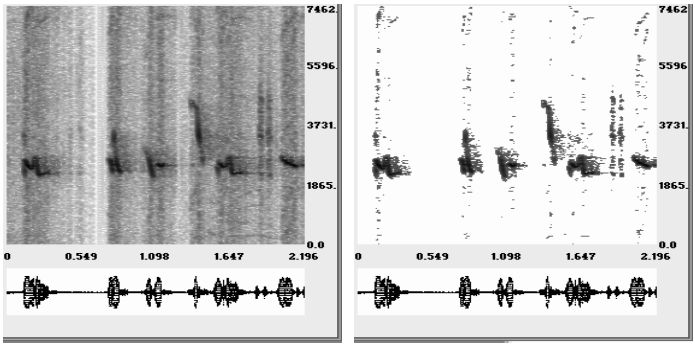
The rationale behind thresholding Gabor transforms is similar to the case of wavelet-based transforms, but the mathematics goes a bit beyond the scope of this primer. Those mathematical details can be found in the preprint [9]. Because this preprint is easily obtained via the Internet (see the web address given in [9]), we shall not provide the mathematical details here. Suffice it to say that it is a preliminary study of a simple technique of Gabor thresholding with some initial comparison to two popular wavelet-based schemes. There are more elaborate wavelet-based methods than the ones used in [9], but, on the other hand, there are also more elaborate Gabor transform methods as well. See the Notes and references for this chapter for further discussion.

We close this section with a couple of examples of denoising real audio. First, in Figure 5.20(a) we show the Blackman-windowed Gabor transform of a recording of an oriole's song. If you listen to the audio file `oriole_song.wav` at the book's website [8], you can hear that this recording is a bit noisy. In Figure 5.20(b) we show the thresholded transform. We computed the inverse of this thresholded transform and saved it as `oriole_song_denoised.wav` at the book's website. Listening to this denoised recording, you should find that the noise has been very effectively removed. In fact there are new textures, previously obscured by the noise, that have been rendered audible. For example, double string vibrations of the oriole's syrinx[9] can now be heard.

Second, we describe the denoising of a recording of a thrush singing in the

[9]The oriole's syrinx is its song generating apparatus, analogous to our larynx.

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 5.20
(a) Spectrogram of oriole's song. (b) Thresholded spectrogram.

wild. See Figure 5.21(a). This example illustrates some of the complications that need to be addressed in denoising real audio signals, which often contain a variety of different types of noise. In addition to Gaussian noise (which cuts off above 8268 Hz, a noise called "pink noise" by audio engineers), there is also a low-pitch rumbling and thumping noise due to electrical humming of the recording equipment and thumping of the microphone during recording. For this denoising it is necessary to modify our basic method. If the Gabor-Blackman thresholding method of [9] is applied there are two problems: (1) there are high-pitch artifacts that are audible (a consequence of the thresholding not completely eliminating some noise that is only slightly above the threshold), (2) the low-pitch rumbling and thumping is not eliminated.

To eliminate the high-pitch artifacts we modify the thresholding procedure by performing *garotte shrinkage*. A simple thresholding, of the type considered so far, alters the magnitudes of Gabor transform values by multiplying each magnitude x by a "hard thresholding" function $H(x)$ defined by

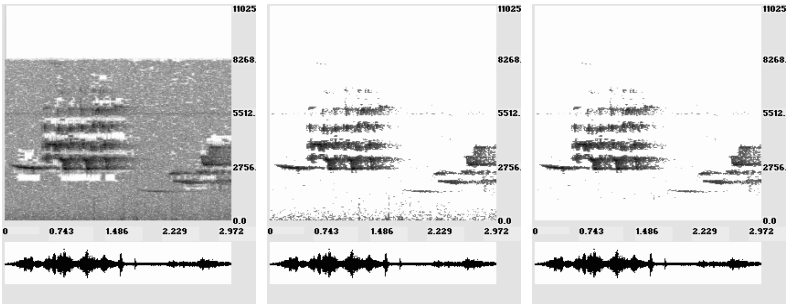
$$H(x) = \begin{cases} 1 & \text{for } x > T \\ 0 & \text{for } 0 \le x \le T. \end{cases}$$

For garotte shrinkage, we instead multiply each Gabor transform magnitude x by the garotte threshold function $G(x)$ defined by

$$G(x) = \begin{cases} 1 - (T/x)^2 & \text{for } x > T \\ 0 & \text{for } 0 \le x \le T. \end{cases}$$

The rationale behind multiplying by $G(x)$ is that if a Gabor transform value's magnitude x is just slightly above the threshold T then it is probably noise and $G(x) \cdot x \approx 0$. But if a Gabor transform value's magnitude x is significantly larger than T, then it is probably a signal-dominated transform value, and the approximation $G(x) \cdot x \approx x$ preserves its magnitude. In Figure 5.21(b)

© 2008 by Taylor & Francis Group, LLC



(a) (b) (c)

FIGURE 5.21
Spectrograms for an audio denoising. (a) Noisy Thrush Song. (b) Garotte thresholded spectrogram. (c) Spectrogram from (b) after high-pass filtering.

we show the spectrogram for the noisy thrush song after garotte shrinkage is applied. An inverse Gabor transform applied at this point would generate an audio signal relatively free of high-pitch artifacts but still suffering from low-pitch rumbling and thumping.

To remove these low-pitch artifacts we performed a "high-pass filtering" of the garotte shrunken transform: retaining only those transform values corresponding to frequencies above 1300 Hz [see Figure 5.21(c)]; and then performing an inverse Gabor transform. The original recording and our denoising, `noisy_thrush.wav` and `denoised_thrush.wav`, are at [8]. Like our first example, we hear in the denoised signal subtle aspects of the thrush song that were obscured by the noise in the original.

5.11 Notes and references

The DFT and the FFT are described in [10] to [13]. Using the DFT for noise removal is described in [13] and [14], and the use of related transforms for compression is discussed in [15].

The detection of abnormal heart signals in ECGs is considerably more complicated than our discussion indicates. Further details on wavelet based methods can be found in the papers [16] and [17].

The construction of the Daub4 scaling numbers and wavelet numbers described in Section 5.6 is adapted from a discussion in [18], where the Daub6 case is examined. The construction of scaling numbers and wavelet numbers in general, based on the Riesz lemma, is described in [19] and [20]. Other methods are described in [21]. There is also an excellent discussion of the construction of the Coif$I$ scaling numbers and wavelet numbers in [22].

A definitive treatment of wavelets from the standpoint of their frequency content is given in [23].

The mathematical theory of Gabor transforms is described with great care in [24]. More mathematical background and applications are treated in [25] and [26]. Lots of primary material, including an archive of important histor-

© 2008 by Taylor & Francis Group, LLC

ical papers and the latest research articles, can be found at the Numerical Harmonic Analysis Group (NuHAG) webpage [27].

For more discussion of spectrograms and music in addition to the treatment in [4], see article [28] and the books [29] and [30]. The book [31] contains a number of insightful articles on the relationship of mathematics and music. A summary of the human perception of music is given in Lady Natasha Spender's article [32] in the excellent compendium [33]. See also the groundbreaking article [34] summarizing Jackendoff and Lerdahl's tonal theory, which is available as a chapter in the computer music book [35].

The notion of music exhibiting repeating patterns at multiple sizes and positions in the time-frequency plane alludes to the mathematical notion of symmetry groups and groups acting on sets. For an elementary introduction to groups in music see Chap. 5 of [36]; a more comprehensive treatment can be found in Chap. 9 of [37]. Some important papers on patterning and mathematical groups in musical structure are [38] to [43]. See also Chapter 21 of the monumental treatise [44].

A tour de force treatment of bird song can be found in the book [45]. David Rothenberg's book [46] contains important philosophical insights, as well as a thoroughly enjoyable introduction to the science of bird song and human and avian musical interaction. The website for Rothenberg's book [47], especially the link *What Musicians Have Done with Bird Song,* contains some beautiful examples of avian-inspired music (including duets of Rothenberg playing along with birds!). More such examples can be found at the *avian music* website [48], including the use of human instruments (voice, sax) to produce time-frequency structures similar in shape and tempo to bird songs. A detailed analysis of the musicality of bird song, including: (1) its relation to classical compositions such as Stravinsky's *Rite of Spring,* (2) transcriptions like we did for the osprey's song, and (3) the use of spectrograms as a generalized musical notation, can all be found in Chapter 5, *Zoomusicology,* of Francois-Bernard Mache's classic book [49].

There is software available for synthesizing music from Gabor transforms. A beautiful example is the MetaSynth program [50].

Further background on Gabor transform denoising is given in Oddvar Christiansen's thesis [51], and in [9]. A state-of-the-art Gabor transform denoising scheme, and its application to denoising speech signals, can be found in [3]. Garotte shrinkage's advantages are discussed in detail in [52]. Some wavelet-based denoising schemes are described in [53] to [55].

1. B. Burke. (1994). The mathematical microscope: waves, wavelets, and beyond. In *A Positron Named Priscilla, Scientific Discovery at the Frontier,* M. Bartusiak, Ed. National Academy Press, 196–235.

2. Y. Meyer. (1993). *Wavelets. Algorithms and Applications.* SIAM, Philadelphia, PA.

3. M. Dörfler. (2002). *Gabor Analysis for a Class of Signals Called Music.* Dissertation, University of Vienna.

© 2008 by Taylor & Francis Group, LLC

4. J.S. Walker and G.W. Don. (2006). Music: a time-frequency approach. Preprint. Available at http://www.uwec.edu/walkerjs/media/TFAM.pdf

5. Audacity software: http://audacity.sourceforge.net/

6. F. Lerdahl and R. Jackendoff. (1983). *A Generative Theory of Tonal Music.* MIT Press, Cambridge, MA.

7. S. Pinker. (1997). *How the Mind Works.* Norton, NY.

8. Book's website: http://www.uwec.edu/walkerjs/primer/

9. J.S. Walker and Y.-J. Chen. (2006). Denoising Gabor transforms. Preprint. Available at

http://www.uwec.edu/walkerjs/media/DGT.pdf

10. W.L. Briggs and V.E. Henson. (1995). *The DFT. An Owner's Manual.* SIAM, Philadelphia, PA.

11. E.O. Brigham. (1978). *The Fast Fourier Transform.* Prentice Hall, Englewood Cliffs, NJ.

12. J.S. Walker. (1988). *Fourier Analysis.* Oxford, New York, NY.

13. J.S. Walker. (1996). *Fast Fourier Transforms, Second Edition.* CRC Press, Boca Raton, FL.

14. J.S. Walker. (1997). Fourier analysis and wavelet analysis. *Notices of the Amer. Math. Soc.,* Vol. 44, 658–670.

15. B.A. Wandell. (1995). *Foundations of Vision.* Sinauer Associates, Sunderland, MA.

16. L. Senhadji, L. Thoraval, and G. Carrault. (1996). Continuous wavelet transform: ECG recognition based on phase and modulus representations and hidden markov models. In A. Aldroubi, M. Unser, Eds., *Wavelets in Medicine and Biology.* CRC Press, Boca Raton, FL, 439–464.

17. M. Akay. (1996). Diagnosis of coronary artery disease using wavelet-based neural networks. In A. Aldroubi, M. Unser, Eds., *Wavelets in Medicine and Biology.* CRC Press, Boca Raton, FL, 513–526.

18. R.S. Strichartz. (1993). How to make wavelets. *The American Math. Monthly,* Vol. 100, 539–556.

19. I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

20. M. Vetterli and J. Kovačević. (1995). *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ.

21. W. Sweldens. (1996). The lifting scheme: a custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis,* Vol. 3, 186–200.

22. C.S. Burrus, R.H. Gopinath, and H. Guo. (1998). *Introduction to Wavelets and Wavelet Transforms, A Primer.* Prentice Hall, Englewood Cliffs, NJ.

23. E. Hernandez and G. Weiss. (1996). *A First Course on Wavelets.* CRC Press, Boca Raton, FL.

© 2008 by Taylor & Francis Group, LLC

24. K. Gröchenig. (2001). *Foundations of Time-Frequency Analysis.* Birkhäuser, Boston, MA.

25. H. Feichtinger and T. Strohmer, Eds. (1998). *Gabor Analysis and Algorithms.* Birkhäuser, Boston, MA.

26. H. Feichtinger and T. Strohmer, Eds. (2002). *Advances in Gabor Analysis.* Birkhäuser, Boston, MA.

27. NuHAG webpage: http://www.univie.ac.at/nuhag-php/home/

28. J.F. Alm and J.S. Walker. (2002). Time-frequency analysis of musical instruments. *SIAM Review,* Vol. 44, 457–476.

29. G. Loy. (2007). *Musimathics: The Mathematical Foundations of Music, Vol. 2.* MIT Press, Cambridge, MA.

30. R. Cogan. (1984). *New Images of Musical Sound.* Harvard University Press, Cambridge, MA.

31. G. Assayag, H. Feichtinger, and J.F. Rodrigues, Eds. (2002). *Mathematics and Music: a Diderot Mathematical Forum.* Springer, New York, NY.

32. N. Spender. (2004). Music, psychology of. Article in *The Oxford Companion to the Mind,* 625–632, Oxford University Press, Oxford, UK.

33. R. Gregory, Ed. (2004). *The Oxford Companion to the Mind,* Oxford University Press, Oxford, UK.

34. F. Lerdahl and R. Jackendoff. (1992). An overview of hierarchical structure in music. In *Machine Models of Music,* S. Schwanauer and D. Levitt, Eds., MIT Press, Cambridge, MA, 289–312.

35. S. Schwanauer and D. Levitt, Eds. (1993). *Machine Models of Music.* MIT Press, Cambridge, MA.

36. L. Harkleroad. (2006). *The Math Behind the Music.* Cambridge University Press, Cambridge, UK.

37. D. Benson. (2006). *Music: A Mathematical Offering.* Cambridge University Press, Cambridge, UK.

38. M. Babbitt. (1961). Set structure as a compositional determinant. *J. of Music Theory,* Vol. 5, 72–94.

39. W. Hodges and R. Wilson. (2002). Musical patterns. In G. Assayag, H.G. Feichtinger, and J.F. Rodrigues, Eds., *Mathematics and Music.* Springer-Verlag, New York.

40. P. Isihara and M. Knapp. (1993). Basic Z_{12} analysis of musical chords. *UMAP Journal,* Vol. 14, 319–348.

41. H. Longuet-Higgins. (1994). Artificial intelligence and musical cognition. *Phil. Trans. Roy. Soc. London,* Vol. A349, 115–131.

42. A. Nestke and T. Noll. (2000). Inner metrical analysis. In J. Haluska (Ed.), *Music and Mathematics,* Tatra Mountains Mathematical Publications, Bratislava.

43. A. Volk. (2004). Metrical investigations in Brahm's symphonies. In E. Lluis Pueboa, G. Mazzola, and T. Noll, Eds., *Perspectives of Mathematical Music Theory,* epOs Music, Osnabrück.

© 2008 by Taylor & Francis Group, LLC

44. G. Mazzola. (2002). *The Topos of Music.* Birkhäuser, Basel.

45. D. Kroodsma. (2005). *The Singing Life of Birds.* Houghton-Mifflin, NY.

46. D. Rothenberg. (2005). *Why Birds Sing: A Journey into the Mystery of Bird Song.* Basic Books, NY.

47. *Why Birds Sing* website: http://www.whybirdssing.com/

48. *Avian music* website: http://www.avianmusic.com/music.html/

49. F.B. Mache. (1993). *Music, Myth and Nature. Contemporary Music Studies, Vol. 6.* Taylor & Francis, London.

50. MetaSynth music software: http://www.uisoftware.com/MetaSynth/

51. O. Christiansen. (2002). Time-frequency analysis and its applications in denoising. Thesis, Department of Informatics, University of Bergen.

52. H.-Y. Gao. (1998). Wavelet shrinkage denoising using the non-negative garrote. *J. of Computational and Graphical Statistics,* Vol. 7, 469–488.

53. D. Donoho and I. Johnstone. (1995). Adapting to unknown smoothness via wavelet shrinkage. *American Statistical Assoc.,* Vol. 90, 1200–1224.

54. H.A. Chipman, E.D. Kolaczyk, and R.E. McCulloch. (1997). Adaptive Bayesian wavelet shrinkage. *J. American Statistical Association,* Vol. 92, 1413–1421.

55. R. Coifman and D. Donoho. (1995). Translation-invariant denoising. In *Wavelets and Statistics,* A. Antoniadis and G. Oppenheim, Eds. Springer, New York, NY.

5.12 Examples and exercises

Section 5.1

Example 5.1.A [**Figure 5.1**] To create (a) we used *Graph/Plot* and plotted

```
2cos(4pi x) + 0.5sin(24pi x)
```

over the interval $[-16, 16]$ using 1024 points. For (b) we then selected *Transform/Fourier* and pressed the *Plot* button. For (c) we opened a new 1-D form, and plotted the function

```
(1 + cos(24 pi x))/(1 + 4x^2)
```

over the interval $[-16, 16]$, after which we created (d) by the method used for (b).

Example 5.1.B [**Figure 5.2**] To get (a) we plotted `del(x)` over the interval $[0, 1024]$ using 1024 points. We then computed a 1-level Coif12 inverse wavelet transform. That produced a plot of the scaling signal $\mathbf{V}_1^1$. To plot the spectrum of $\mathbf{V}_1^1$, we chose *Transform/Fourier,* selected the options *Power sp.* and $[0, L] \to [-A, A]$, and unchecked the box labelled *Periodic, endpoint averaged.* For (b) we did the same work, except we first plotted `del(x-512)`.

Example 5.1.C We show in Figure 5.22 the graphs of the spectra of the Coif12 scaling signal $\mathbf{V}_{17}^1$ and wavelet $\mathbf{W}_{17}^1$. These spectra were produced as in Example 5.1.B, using `del(x-16)` and `del(x-512-16)`. Notice how they match the spectra of $\mathbf{V}_1^1$ and $\mathbf{W}_1^1$ shown in Figure 5.2.

© 2008 by Taylor & Francis Group, LLC

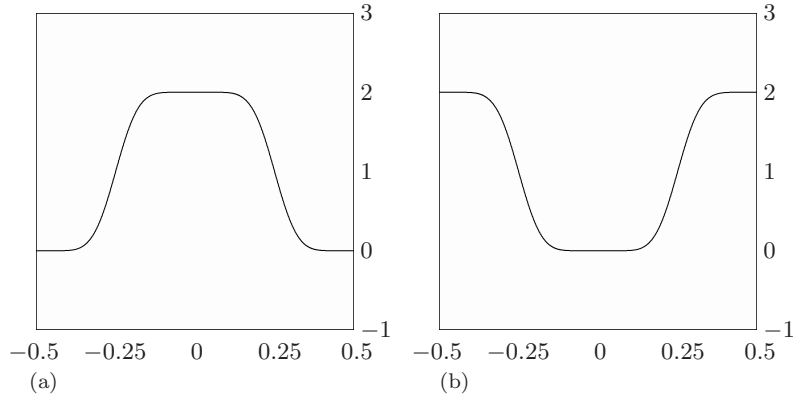


FIGURE 5.22
(a) Spectrum of Coif12 scaling signal $\mathbf{V}_{17}^1$. (b) Spectrum of Coif12 wavelet $\mathbf{W}_{17}^1$.

5.1.1$_{\mathbf{s}}^{\mathbf{c}}$ Produce a graph of

$$3\sin(8\pi x) - 2\cos(16\pi x)$$

over the interval $[-16, 16]$ using 1024 points, and then produce a plot that displays the frequency content of this function.

5.1.2$^{\mathbf{c}}$ Produce a graph of

$$2\cos(12\pi x) + 8\sin(24\pi x)$$

over the interval $[-16, 16]$ using 1024 points, and then produce a plot that displays the frequency content of this function.

5.1.3$_{\mathbf{s}}^{\mathbf{c}}$ Plot spectra of the Coif12 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.4$^{\mathbf{c}}$ Plot spectra of the Coif18 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.5$^{\mathbf{c}}$ Plot spectra of the Daub12 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.6$^{\mathbf{c}}$ Plot spectra of the Haar scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

Section 5.2

5.2.1$_{\mathbf{s}}$ Prove the Linearity Property of the DFT.

5.2.2$_{\mathbf{s}}$ Prove the Periodicity Property of the DFT.

5.2.3$_{\mathbf{s}}$ Prove the Inversion Property of the DFT.

5.2.4 Prove Parseval's Equality for the DFT.

5.2.5$_{\mathbf{s}}$ Find the z-transform of $\mathbf{f} = (1, 1, 1, 1)$. What are its roots?

5.2.6 Find the z-transform of $\mathbf{f} = (1, 0, 1, 0, 1, 0, 1, 0)$. What are its roots?

5.2.7 Prove that $\mathcal{T}_k \circ \mathcal{T}_m = \mathcal{T}_{k+m}$. Also prove that $\mathcal{T}_{-k} = \mathcal{T}_k^{-1}$.

© 2008 by Taylor & Francis Group, LLC

Section 5.3

Example 5.3.A [Figure 5.3] To produce the graph in (a) we first plotted

```
(1 + cos(24 pi x))/(1 + 4x^2)
```

over the interval $[-16, 16]$ using 1024 points. We then selected *Series/Wavelet* and chose a 1-level Coif12 wavelet with option *Ascending terms* and entered 512 for the number of terms. Plotting that series produced the graph shown in (a). To produce the graph in (b) we performed a Fourier transform of the graph in (a). We obtained the graph in (c) by subtracting the graph in (a) from the original function's graph. The graph in (d) is the Fourier transform of the graph in (c).

Example 5.3.B [Figure 5.4] To obtain the graphs in (a) we computed power spectra of scaling signals $\mathbf{V}_1^k$ and multiplied each power spectrum by 2^{-k} for $k = 1$ to $k = 4$. For example, to plot $\mathbf{V}_1^3$ we graphed `del(x)` over $[0, 1024]$ using 1024 points, and then computed a 3-level inverse Coif12 transform. Plotting the power spectrum and multiplying by 2^{-3} produced the 3rd graph from the top in (a). Similar work was done to produce the graphs in (b) except that wavelets $\mathbf{W}_1^k$ were used. For instance, to plot the graph of $\mathbf{W}_1^3$ we plotted `del(x-128)` over $[0, 1024]$ using 1024 points, and then computed an inverse 3-level Coif12 transform.

5.3.1$_s^c$ In Figure 5.23(a) we show the graph of the function

$$\frac{1+\cos(16\pi x)}{1+4x^2} + \frac{1+\sin(24\pi x)}{1+4(x-8)^2} + \frac{1-\cos(8\pi x)}{1+4(x+8)^2}$$

over the interval $[-16, 16]$ using 1024 points, and in (b) we show the graph of its DFT. Reproduce these graphs and identify the portions of the DFT graph that correspond to transforms of the 3-level Coif12 averaged signal $\mathbf{A}^3$, and detail signals $\mathbf{D}^3$, $\mathbf{D}^2$, $\mathbf{D}^1$.

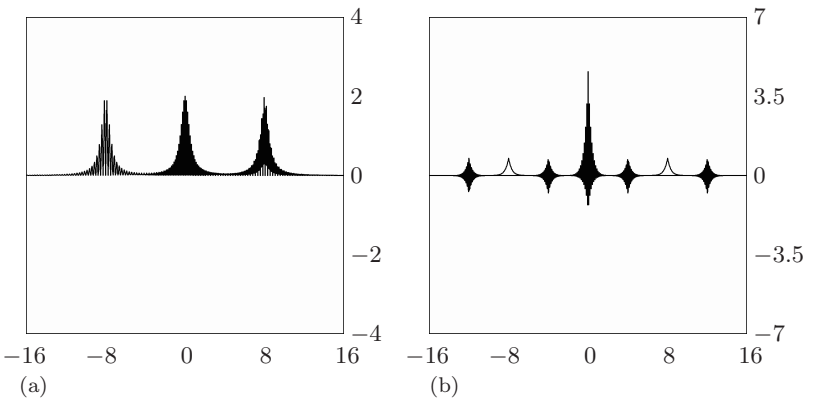


FIGURE 5.23
(a) Graph of function from Exercise 5.3.1. (b) DFT of that function.

© 2008 by Taylor & Francis Group, LLC

5.3.2^{c} Graph the function

$$\sum_{k=1}^{4} \frac{1 + 4\cos(2\pi 3 \cdot 2^{k-1}x)}{1 + 4(x - (-40 + 16k))^2}$$

over the interval $[-32, 32]$ using 4096 points, and plot its DFT. For a Coif12 wavelet, identify which portions of the DFT correspond to transforms of the signals $\mathbf{A}^4$, $\mathbf{D}^4$, $\mathbf{D}^3$, $\mathbf{D}^2$, $\mathbf{D}^1$.

5.3.3 Prove formula (5.29).

Section 5.4

Example 5.4.A [**Figure 5.5**] The graph of the top signal was obtained by plotting

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k-1)>.5))
+.1sin(12pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x-2
```

over the interval $[-5, 5]$ using 1024 points. The middle signal was obtained by then plotting

```
100u(u-.1)(u+.2)(-.2<u<.1) \u=x
```

over the same interval with the same number of points. To obtain the bottom graph we selected *Convolve/Pair Correlation*, and entered 2 for *Graph 1* and 1 for *Graph 2* with the *Normalize* option selected, and after plotting this normalized correlation, we then plotted the graph of `g3(x)(g3(x)>.9)` to show those correlation values that exceeded 0.9.

5.4.1^{c} Compute a graph showing the values of the normalized correlation of the following two graphs, over the interval $[-5, 5]$ using 1024 points:

$$\mathbf{g}: \quad 2(-3 < x < -2) + (1 - \text{abs}(x))(\text{abs}(x) < 1) - 1(3 < x < 4)$$

$$\mathbf{f}: \quad 2(-.5 < x < .5)$$

Explain why the position of the maximum of 1 for the normalized correlation occurs where it does, and why the minimum of -1 occurs where it does.

5.4.2^{c} Compute a graph showing the values that exceed 0.90 of the normalized correlation of the following two graphs, over the interval $[-5, 5]$ using 1024 points:

$$\begin{aligned}\mathbf{g}: \quad & \text{sumk}(100u(u - .1)(u + .2)(-.2 < u < .1)(\text{abs}(k - 1) > .5)) \\ & + .1\cos(14\pi v)(-.2 < v < .1) \\ & \backslash u = x - 2k \backslash k = -2, 2 \backslash v = x - 2\end{aligned}$$

$$\mathbf{f}: \quad 100u(u - .1)(u + .2)(-.2 < u < .1) \backslash u = x$$

© 2008 by Taylor & Francis Group, LLC

Section 5.5

Example 5.5.A [**Figure 5.6**] To produce this figure you first load the image `P.bmp` and then the image `PQW.bmp` and that displays Gr 1 and Gr 2. To produce Gr 3 you select *Convolve/Pair correlation* and check the box entitled *Normalize.* You also enter 1 for Graph 1, and 1 for Graph 2. After plotting the normalized correlation as the third graph and then plotting `(g3 > .9)` you obtain the tiny white pixel at the center shown in Gr 3. To obtain the image in Gr 4 you proceed in a similar way, except you enter 1 for Graph 1 and 2 for Graph 2 when computing the normalized correlation.

Example 5.5.B [**Figure 5.7**] To obtain the graphs in (a) you load successively the images `Elaine_face.bmp`, `boat.bmp`, `Elaine.bmp` and `Zelda.bmp`. To obtain the images in (b) you perform normalized correlations with `Elaine_face.bmp` as Graph 1, and each of the images in (a) as Graph 2, and you also plot the values exceeding 0.9 for each of these normalized correlations.

Example 5.5.C [**Figure 5.8**] To obtain (a) you proceed as follows. First, clip out the image for `Elaine_face.bmp` so that it is displayed in a new window. Then, perform a 1-level Coif12 transform of `Elaine_face.bmp`, then graph `g2(1-(x<0)(y<0))` and then inverse transform Gr 3 to get $\mathbf{D}^1$. [To see $\mathbf{D}^1$, you change the *View* style to *Log* and *Grey (+/-).*] You then plot

```
g4((x^2 + y^2) < c^2) \c = 0.15
```

to remove extraneous edges at the boundary of the disc containing edges of Elaine's face. Then repeat this work (without the boundary removal step) for each of the images `boat.bmp`, `Elaine.bmp`, and `Zelda.bmp`, copying and pasting the $\mathbf{D}^1$ images into the form containing the face version of $\mathbf{D}^1$, and computing normalized correlations with thresholding at 0.9.

5.5.1$^{\mathbf{c}}$ Do a normalized correlation, and retain only values exceeding 0.9, for the image `2DCorr_a.bmp` of the letter *a* within the portion of text in the image `2DCorr_text.bmp`. Verify that all instances of the letter *a* and their positions are detected correctly.

5.5.2$^{\mathbf{c}}$ Do a normalized correlation, and retain only values exceeding 0.9, for the image `2DCorr_u.bmp` of the letter *u* within the portion of text in the image `2DCorr_text.bmp`. Verify that all instances of the letter *u of the same size* are detected correctly. **Note:** Some larger size versions of `u` are not detected. This provides evidence that our brains use much more sophisticated methods of pattern matching, valid across a range of sizes, not just a single fixed size.

5.5.3$_{\mathbf{s}}$ In the previous two exercises, the text and the individual letters were white on a black background. Explain how to handle letter detection if the text and letter are both black on a white background. Do you see any relation to this problem and vision?

5.5.4$^{\mathbf{c}}$ Apply the edge correlation method to detecting the presence of `Barb_face` within the image `Barb.bmp` and its lack of presence within the images `Zelda.bmp` and `Elaine.bmp`. [*Note:* To observe the detection pixel, view the images at *Large* size.]

© 2008 by Taylor & Francis Group, LLC

5.5.5$^{\mathbf{c}}$ Repeat the previous exercise, but use edge correlations based on the 3$^{\text{rd}}$ level Coif12 trends for the images.

Section 5.6

5.6.1 Show that for $P(\theta) = e^{i\pi}\cos\pi\theta$, we obtain the Haar scaling numbers $\alpha_1 = \alpha_2 = 1/\sqrt{2}$ and wavelet numbers $\beta_1 = 1/\sqrt{2}$, $\beta_2 = -1/\sqrt{2}$.

5.6.2 Show that (5.67) implies (5.68).

5.6.3$_{\mathbf{s}}$ Use the method of this section to derive the Daub6 scaling numbers and wavelet numbers.

5.6.4 Generalize the method of this section to the biorthogonal case.

Section 5.7

Example 5.7.A [Figure 5.9] To produce (a) we plotted the function

$$e^{-400(t-0.2)^2}\sin 1024\pi t + e^{-400(t-0.5)^2}\cos 2048\pi t + e^{-400(t-0.7)^2}(\sin 512\pi t - \cos 3072\pi t)$$

over the interval $[0, 1]$ using 8192 points. The DFT in (b) was then produced by choosing *Transform/Fourier* and selecting `[0,L]->[-A,A]` for *Interval Type*. To produce (c) we selected *Analysis/Spectrograph* from the window containing the graph in (a). When the spectrogram window opened we then plotted the spectrogram with *Blackman* specified as the window. We produced (d) by selecting *Boxcar* for the window in the spectrogram.

Example 5.7.B [Figure 5.10] The graph in (a) was created by opening a new 1D-form, then right-clicking on the graph region followed by selection of *Load/Sound file*. We then selected the sound file `flute_clip.wav`. The graph in (b) was created by plotting several shifts of the Blackman window, and (c) was obtained by multiplying the signal in (a) by the central window displayed in (b).

Example 5.7.C [Figure 5.11] This figure was obtained by plotting the formulas for the Hanning and Blackman window functions, using $\lambda = 1$, over the interval $[-0.5, 0.5]$ using 1024 points.

5.7.1$^{\mathbf{c}}_{\mathbf{s}}$ Plot the function

$$e^{-400(t-0.2)^2}\sin 2048\pi t + e^{-400(t-0.5)^2}\cos 512\pi t + e^{-400(t-0.7)^2}(\sin 1024\pi t - \cos 3072\pi t)$$

over the interval $[0, 1]$ using 8192 points, and then compute its Hanning and Blackman windowed spectrograms. Do you observe any differences?

5.7.2$^{\mathbf{c}}$ Plot the function

$$e^{-400(t-0.2)^2}\sin 512\pi t + e^{-400(t-0.5)^2}\cos 2048\pi t + e^{-400(t-0.7)^2}(\sin 512\pi t + 0.5\cos 1024\pi t - \cos 3072\pi t)$$

© 2008 by Taylor & Francis Group, LLC

over the interval $[0, 1]$ using 8192 points, and then compute its Hanning and Blackman windowed spectrograms.

5.7.3^{c} Load the signal `greasy.wav` and compute its Blackman windowed spectrogram.

5.7.4^{c} Load the signal `Chong's 'Bait'.wav` and compute its Blackman windowed spectrogram.

Section 5.8

Example 5.8.A [**Figure 5.12**] The spectrogram in (a) was generated by loading the sound file `piano_clip.wav` and then computing a Blackman windowed spectrogram. For (b) we plotted the formula `sin[8192(pi/3)x^3]` over the interval $[0, 1]$ using 8192 points and then computed a Blackman windowed spectrogram.

Example 5.8.B [**Figure 5.13**] The spectrum in (a) was computed in the following way. In the *spectrogram* window containing the Blackman windowed spectrogram of the piano notes (see last example), we selected *View* and then chose *View cursor coordinates.* We then moved the mouse cursor over the spectrogram until the tip of the mouse cursor displayed 0.6 as its first coordinate, and then right-clicked and chose *Vertical slice* from the popup menu. That produced the real and imaginary parts of the FFT corresponding to the vertical slice along $t = 0.6$ in the spectrogram. To get the spectrum's plot, we graphed the function

```
sqr(g1(x)^2 + g2(x)^2)
```

and then clipped out graph 3, and changed the display style to *Lines* with the X and Y ranges shown in the figure. To produce (b) we used the same steps, except that we right-clicked on the spectrogram when the first coordinate was 1.115.

Example 5.8.C [**Figure 5.14**] The spectrogram in this figure was created by juxtaposing two spectrograms. We began by loading the file `firebird_clip2.wav`, which loaded into an *Audio editor* window. We then left-clicked at about a quarter of the way from the left end of the signal (at about 54000 for the *Line 1* reading) to create a beginning clip-line, followed by a click toward the middle of the signal to create an end click line. We then right-clicked on the selected region and chose *Clip.* By clicking on the *Analyze* button we opened the clipped signal within a new 1D-form and then computed a Blackman windowed spectrogram. This spectrogram is the left half of the spectrogram shown in the figure.

To produce the right half, we right-clicked on the right clip line and selected *Move clip region.* The clipped signal shown in the small box on the right of the *Audio editor* is automatically updated with this new clipping. We then clicked the *Analyze* button and created a Blackman windowed spectrogram, which is the right half of the spectrogram shown in the figure.

Example 5.8.D [**Figure 5.15**] To create the spectrogram in (a) we loaded the sound file `Chinese_folk_Music.wav` and computed a Blackman windowed spectrogram. The zooming in shown in (b) was computed in the following way (the mathematics underlying the following procedure is discussed in Sections 6.3 and 6.4).

© 2008 by Taylor & Francis Group, LLC

From the menu for the original sound signal we selected *Analysis* and chose *Scalogram.* In the *scalogram* window that opened, we chose to compute a *Gabor (complex)* scalogram, using the following settings:

```
Octaves: 2      Voices: 128

Width: 0.25     Freq.: 125
```

and then clicked on the *Plot* button to compute the scalogram. Once the scalogram was plotted, we then selected *View/Display style* and selected *Log (global)* for the *Magnitude* setting.

Example 5.8.E [**Figure 5.16**] These spectrograms were created by loading the sound files `osprey_song.wav` and `oriole_song.wav` and computing Blackman windowed spectrograms.

5.8.1$^{\mathbf{c}}$ Analyze the oriole's song from the recording `oriole_song.wav`.

5.8.2$^{\mathbf{c}}_{\mathbf{s}}$ Use AUDACITY to create a spectrogram of the passage from the classical Chinese folk song recorded in the file `Happiness_clip.wav`, and use the Multiresolution Principle to analyze its musical qualities. Details on how to use AUDACITY can be obtained by selecting the link *Document on using FAWAV and Audacity* available at the website `http://www.uwec.edu/walkerjs/tfam/`

5.8.3$^{\mathbf{c}}$ Analyze the music in the recording `BA_passage.wav` from the song *Buenos Aires.*

Section 5.9

Example 5.9.A [**Figure 5.17**] To create (a) we loaded `oriole_whistle.wav` and performed a Blackman windowed spectrogram. We created (b) by selecting *Graph/Plot* from the spectrogram window displaying (a), and then clicking on the *Load* button to load the formula file `synthetic_oriole_whistle.uf2`, and then plotting this formula. (The sound signal corresponding to this spectrogram is generated by selecting *Graph/Inverse.* It should be played at the same sampling rate and bit-rate as the oriole whistle.) For (c), we selected *Graph/Restore* from the spectrogram menu, then proceeded as for (b) except that we plotted the formula from the file: `a_synthetic_bird_song.uf2`. *Note:* Make sure you play this latter signal at 16 bits.

5.9.1$^{\mathbf{c}}$ Model the spectrogram of `house_wren_chirp.wav` and compute a synthesized mimic of this bird call.

5.9.2$^{\mathbf{c}}_{\mathbf{s}}$ Compute a spectrogram of the recorded sound `warbler.wav`. Analyze this spectrogram for its musical qualities using the principles discussed in the previous section.

5.9.3$^{\mathbf{c}}$ Model the spectrogram of `warbler.wav` and compute a synthesized mimic of this bird song.

5.9.4$^{\mathbf{c}}_{\mathbf{s}}$ For the signal `original clip from Firebird Suite.wav`, selectively amplify the harp glissando.

© 2008 by Taylor & Francis Group, LLC

5.9.5[c] Using the Multiresolution Principle, synthesize your own bird song and/or musical passage.

5.9.6 [Multi-window Gabor transforms] A Gabor transform does not have to utilize a fixed window. Suppose that we have a set of windows $\{w_m(t_k-\tau_m)\}_{m=0}^M$ satisfying

$$A \le \sum_{m=0}^{M} w_m^2(t_k-\tau_m) \le B$$

for two positive constants A and B. Prove that the multi-window Gabor transform of $\{g(t_k)\}$:

$$\{\mathcal{F}\{g(t_k)w(t_k-\tau_m)\}\}_{m=0}^M$$

has an inverse. *Note:* These windows may be chosen in an adaptive, signal dependent way. See the discussion of local cosine series in Section 6.6 (p. 243).

Section 5.10

Example 5.10.A [Figure 5.18] To perform the denoising illustrated in the figure we proceeded as follows. First, we graphed the following multiple of the chirp formula given in the text:

```
c sin(8192 (pi/3)x^3) \c = 7/.695750610959927
```

over the interval $[0,1]$ using 8192 points. The constant c is chosen so that the standard deviation of the values of this signal is 7 (that is the standard deviation set by Donoho for the test signals he created, which are widely used in benchmarking denoising algorithms). After graphing this chirp signal, we then added Gaussian random noise of standard deviation $\sigma = 1$ to it by plotting the formula

```
c sin(8192 (pi/3)x^3) + rang(0)
\c = 7/.695750610959927
```

The MSE for the noisy chirp compared to the original chirp is then calculated by choosing *Norm difference* and selecting the *Power norm* option with power 2. That gives the Root Mean Square (RMS) error; the MSE is the square of that RMS value.

The Blackman windowed spectrogram in (a) was created from this noisy chirp signal. Its thresholded spectrogram in (b) was created by choosing *Graph/Denoise* from the spectrogram menu and plotting the formula that FAWAV automatically supplies. [The theory that explains how that denoising formula is created is discussed in the paper *Denoising Gabor transforms* (reference [9] for this chapter). To get the MSE error for denoising, you then select *Graph/Invert* from the spectrogram menu and copy and paste the resulting signal in to the window containing the original chirp signal, and *square the RMS error* (computed between graphs 1 and 3).

Example 5.10.B [Figure 5.19] To create the graph shown in (a) you plot the formula `bumps.uf1`, available from the `Primer_Formulas.zip` archive from the *Audio, Images, Formulas* link at the book's website. The plotting is done over the interval $[0,1]$ using 8192 points. You then calculate that signal's standard deviation (which my computer gave as 1.53952824076645) and plot the following signal

```
7g1(x)/c \c = 1.53952824076645
```

© 2008 by Taylor & Francis Group, LLC

After removing graph 1 (right-click on the graph region and select *Remove graph* and specify graph 1 for removal), you are left with the plot of Donoho's test function *Bumps* shown in (a). The noisy signal in (b) is plotted as in the previous example (using `g1(x)+rang(0)`). The denoising in (c) is obtained by selecting *Graph* and then *Denoise (Gabor)* and specifying graph 2. A spectrogram window opens up with a formula for plotting the thresholded spectrogram (it is not automatically computed because you are able to modify the formula for more advanced denoising such as garotte shrinkage, see Example 5.10.D below). After plotting the thresholded spectrogram and selecting *Graph/Invert* you obtain the denoised signal shown in (c). The MSEs were computed as described in the previous example.

Example 5.10.C [Figure 5.20] To produce the spectrograms in Figure 5.20, you load the sound file `oriole_song.wav` and compute its Blackman windowed Gabor transform to get (a), then select *Graph/Denoise* to plot the thresholded spectrogram (b). The denoised signal was produced by selecting *Graph/Invert*. We then saved this signal as `oriole_song_denoised.wav` by right-clicking on it and selecting *Save/Sound file*.

Example 5.10.D [Figure 5.21] To obtain the spectrogram in (a) you load the audio file `noisy_thrush.wav` and plot a Blackman windowed spectrogram. If you select *Graph/Denoise* then the automatically generated formula for thresholding is

```
(g1)(g1 > c)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

If you plot this formula and then select *Graph/Invert* you will produce a denoising that suffers from high-pitched artifacts and low-pitch rumbling and thumping. The garotte shrinkage shown in (b) is performed by selecting *Graph/Restore*, followed by *Graph/Denoise*, and then modifying the automatically generated formula to obtain:

```
(g1)(g1 > c)(1-(c/g1)^2)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

Plotting this formula produces (b). If you then select *Graph/Invert* you will create a sound signal that no longer suffers from high-pitched artifacts, but still has low-pitch rumbling and thumping. These latter noises are removed by a high-pass filtering, illustrated in (c). To obtain the spectrogram in (c), you select *Graph/Restore* from the spectrogram's menu, followed by *Graph/Denoise*. You then modify the automatically generated formula to obtain:

```
(g1)(g1 > c)(1-(c/g1)^2)(y > 1300)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

Plotting this formula produces (c). If you then select *Graph/Invert* you will create a sound signal that is relatively noise-free, no longer suffering from either high-pitched artifacts or low-pitch rumbling and thumping.

Example 5.10.E In this example we consider a challenging denoising of a real audio signal. The noisy recording is the audio file `Chinese_Folk_Music.wav`. We show its Blackman windowed spectrogram in Figure 5.24(a). Its thresholded spectrogram is graphed by plotting the following function (obtained by selecting *Graph/Denoise* from the spectrogram menu):

© 2008 by Taylor & Francis Group, LLC

```
(g1)(g1 > c)
\c=.55*sqr(2log(1024))*(355.195793541264)*sqr(1024)
```

See Figure 5.24(b). Almost all of the noise has been removed. Unfortunately,

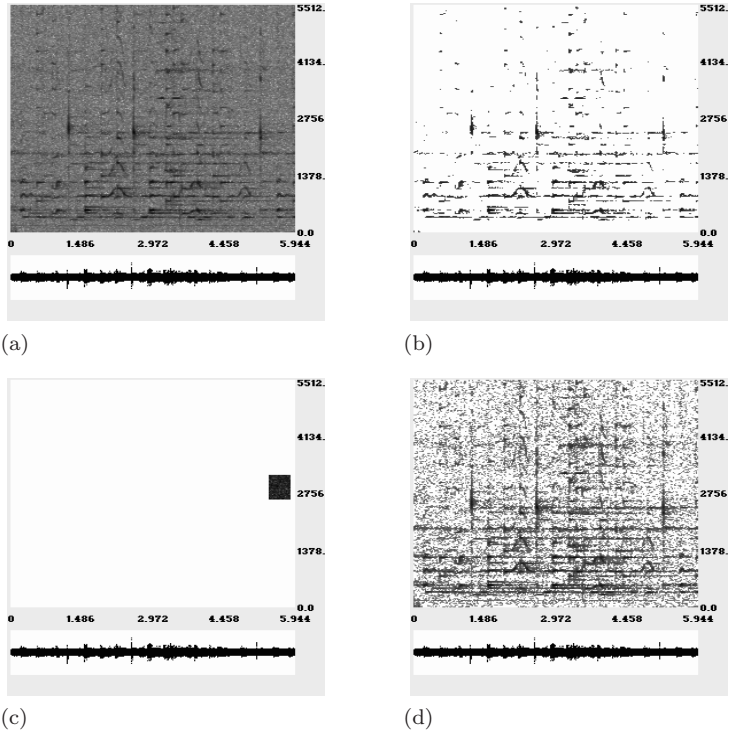


FIGURE 5.24
Denoising Chinese folk music. (a) Noisy signal spectrogram. (b) Thresholded Gabor transform. (c) Small region of noise from (a). (d) Garrote shrinkage applied to Gabor transform for (a) with a new standard deviation.

however, some signal-dominated values are lost (especially in the higher frequencies). When the denoised signal obtained from the thresholded Gabor transform is played it sounds "muddy" due to loss of some of the high frequency content of the music.

To fix this problem, which is due to the rather high estimate of the standard deviation ($\sigma \approx 355$), we need a new, lower estimate. By examining the spectrogram in Figure 5.24(a) we see that there is a region of the time-frequency plane, shown in Figure 5.24(c), that is mostly noise. To obtain the plot in (c) we used the function:

```
g1(5.302<x<5.746)(2680<y<3263)
```

After performing an inverse Gabor transform, we then changed the X-range to $5.302, 5.746$ and calculated the standard deviation of the noisy signal. We got a new estimate of the standard deviation: $\sigma \approx 93$. Returning to the spectrogram window and restoring the noisy spectrogram of the full signal, we then plotted the spectrogram shown in (d) using the following garotte shrinkage:

© 2008 by Taylor & Francis Group, LLC

```
(g1)(g1 > c)(1-(c/g1)^2)\c=.55*sqr(2log(1024))*(93)*sqr(1024)
```

Although this spectrogram appears noisy, when it is inverted the resulting denoised signal sounds noise-free, and more sharply defined due to better preservation of the high-frequency content of the music. We have saved this denoising as `denoised_Chinese_music_garotte.wav`.

5.10.1[c] Denoise the noisy recording `Chinese_Musical_Passage_Clip_b.wav`.

5.10.2[c] Denoise the noisy recording `Dan's 'Bait'.wav`.

5.10.3[c] Denoise the noisy recording `bobolink.wav`. [Note: garotte shrinkage will produce a *Run error* due to division by 0 in `g1(g1>c)(1-(c/g1)^2)`. Use the modified expression `g1(g1>c)(1-(c/(g1+(g1<c/2))^2))` to avoid division by 0.]

© 2008 by Taylor & Francis Group, LLC

Chapter 6

Beyond wavelets

When you have only one way of expressing yourself, you have limits that you don't appreciate. When you get a new way to express yourself, it teaches you that there could be a third or a fourth way. It opens up your eyes to a much broader universe.

—David Donoho[1]

In this chapter we explore some additional topics that extend the basic ideas of wavelet analysis. We first describe *wavelet packet transforms,* which sometimes provide superior performance beyond that provided by wavelet transforms. Then we discuss *continuous wavelet transforms* which are particularly useful for tackling problems in signal recognition, and for performing finely detailed examinations of the structure of signals—we shall look at examples from ECGs, speech, and music. In the Notes and References section we provide a guide to the literature for more topics which, for reasons of space, we could not discuss.

6.1 Wavelet packet transforms

Wavelet packet transforms are a profound generalization of wavelet transforms. In this section we provide a brief introduction to the mathematics of wavelet packet transforms, and in the next section examine some of their applications.

[1]Donoho's quote is from [1].

© 2008 by Taylor & Francis Group, LLC

All wavelet packet transforms are calculated in a similar way. Therefore we shall concentrate initially on the Haar wavelet packet transform, which is the easiest to describe. One type of Haar wavelet packet transform is the *Walsh transform.* A Walsh transform is calculated by performing a 1-level Haar transform *on all subsignals,* both trends *and* fluctuations.

For example, consider the signal $\mathbf{f}$ defined by

$$\mathbf{f} = (2, 4, 6, 8, 10, 12, 14, 16). \tag{6.1}$$

A 1-level Haar transform and a 1-level Walsh transform of $\mathbf{f}$ are identical, producing the following signal:

$$(3\sqrt{2}, 7\sqrt{2}, 11\sqrt{2}, 15\sqrt{2} \,|\, -\sqrt{2}, -\sqrt{2}, -\sqrt{2}, -\sqrt{2}). \tag{6.2}$$

A 2-level Walsh transform is calculated by performing 1-level Haar transforms on both the trend and the fluctuation subsignals, as follows:

$$(3\sqrt{2}, 7\sqrt{2}, 11\sqrt{2}, 15\sqrt{2}) \overset{\mathbf{H}_1}{\longmapsto} (10, 26 \,|\, -4, -4)$$
$$(-\sqrt{2}, -\sqrt{2}, -\sqrt{2}, -\sqrt{2}) \overset{\mathbf{H}_1}{\longmapsto} (-2, -2 \,|\, 0, 0).$$

Hence the 2-level Walsh transform of the signal $\mathbf{f}$ is the following signal:

$$(10, 26 \,|\, -4, -4 \,|\, -2, -2 \,|\, 0, 0). \quad \text{[2-level Walsh]} \tag{6.3}$$

It is interesting to compare this 2-level Walsh transform with the 2-level Haar transform of the signal $\mathbf{f}$. The 2-level Haar transform of $\mathbf{f}$ is the following signal:

$$(10, 26 \,|\, -4, -4 \,|\, -\sqrt{2}, -\sqrt{2}, -\sqrt{2}, -\sqrt{2}). \quad \text{[2-level Haar]} \tag{6.4}$$

Comparing this Haar transform with the Walsh transform in (6.3), we see that the Walsh transform is slightly more compressed in terms of energy, since the last two values of the Walsh transform are zeros. We could, for example, achieve 25% lossless compression of the signal $\mathbf{f}$ by discarding the two zeros from its 2-level Walsh transform, but we could not discard any zeros from its 2-level Haar transform. Another advantage of the 2-level Walsh transform is that it is more likely that *all* of its non-zero values would stand out from a random noise background, because these values have larger magnitudes than the values of the 2-level Haar transform.

A 3-level Walsh transform is performed by calculating 1-level Haar transforms on each of the four subsignals that make up the 2-level Walsh transform. For example, applying 1-level Haar transforms to each of the four subsignals of the 2-level Walsh transform in (6.3), we obtain

$$(10, 26) \overset{\mathbf{H}_1}{\longmapsto} (18\sqrt{2} \,|\, -8\sqrt{2}),$$
$$(-4, -4) \overset{\mathbf{H}_1}{\longmapsto} (-4\sqrt{2} \,|\, 0),$$
$$(-2, -2) \overset{\mathbf{H}_1}{\longmapsto} (-2\sqrt{2} \,|\, 0),$$
$$(0, 0) \overset{\mathbf{H}_1}{\longmapsto} (0 \,|\, 0).$$

© 2008 by Taylor & Francis Group, LLC

Hence the 3-level Walsh transform of the signal $\mathbf{f}$ in (6.1) is

$$(18\sqrt{2}\,|\,-8\sqrt{2}\,|\,-4\sqrt{2}\,|\,0\,|\,-2\sqrt{2}\,|\,0\,|\,0\,|\,0). \quad \text{[3-level Walsh]} \tag{6.5}$$

Here, at the third level, the contrast between the Haar and Walsh transforms is even sharper than at the second level. The 3-level Haar transform of this signal is

$$(18\sqrt{2}\,|\,-8\sqrt{2}\,|\,-4,-4\,|\,-\sqrt{2},-\sqrt{2},-\sqrt{2},-\sqrt{2}). \quad \text{[3-level Haar]} \tag{6.6}$$

Comparing the transforms in (6.5) and (6.6) we can see, at least for this particular signal $\mathbf{f}$, that the 3-level Walsh transform achieves a more compact redistribution of the energy of the signal than the Haar transform.

In general, a wavelet packet transform is performed by calculating a particular 1-level wavelet transform for some or all of the subsignals of the preceding level. For simplicity, we shall concentrate on the case where each subsignal is transformed at every level (see Section 6.7 for discussion of other cases). For example, a 3-level Daub4 wavelet packet transform would be calculated in the same way as a 3-level Walsh transform, but with 1-level Daub4 wavelet transforms being used instead of 1-level Haar transforms. Because all of the 1-level wavelet transforms that we have discussed enjoy the conservation of energy property (nearly so for Daub 9/7) and have inverses, it follows that all of their wavelet packet transforms also enjoy the conservation of energy property (nearly so for Daub 9/7) and have inverses. What this implies is that our discussions of compression and denoising of signals in Chapters 2 and 3 apply, essentially without change, to wavelet packet transforms. In particular, the threshold method is still the basic method for compression and noise removal with wavelet packet transforms.

In two dimensions, a wavelet packet transform is performed by adopting the same approach that we used in one dimension. First, a 1-level wavelet transform is performed on the 2D image. Then, to compute a 2-level wavelet packet transform, this 1-level wavelet transform is applied to each of the four subimages—$\mathbf{a}^1$, $\mathbf{d}^1$, $\mathbf{h}^1$, and $\mathbf{v}^1$—from the first level. This produces 16 subimages that constitute the 2-level wavelet packet transform of the image. To compute a 3-level wavelet packet transform, the 1-level wavelet transform is applied to each of these 16 subimages, producing 64 subimages. This process continues in an obvious manner for higher level wavelet packet transforms.

Because of the great similarity between wavelet transforms and wavelet packet transforms, we shall now end our discussion of the mathematics of these transforms, and turn to a discussion of a few applications.

6.2 Wavelet packet transforms—applications

In this section we shall discuss a few examples of applying wavelet packet transforms to audio and image compression. While wavelet packet transforms

© 2008 by Taylor & Francis Group, LLC

can be used for other purposes, such as noise removal, because of space limitations we shall limit our discussion to the topic of compression.

For our first example, we shall use a Coif30 wavelet packet transform to compress the audio signal *greasy,* considered previously in Section 3.5. In that section we found that a 4-level Coif30 wavelet transform—with trend values quantized at 8 bpp and fluctuations quantized at 6 bpp, and with separate entropies computed for all subsignals—achieved a compression of *greasy* requiring an estimated 13,107 bits. That is, this compression required an estimated 0.8 bpp (instead of 8 bpp in the original). However, if we use a 4-level Coif18 wavelet packet transform and quantize in the same way, then the estimated number of bits is 9,175, i.e., 0.56 bpp. This represents a significant improvement over the wavelet transform.

In several respects—in bpp, in RMS Error, and in total number of significant values—the wavelet packet compression of *greasy* is nearly as good as or significantly better than the wavelet transform compression. See Table 6.1. Our next example, from the field of image compression, is even more striking.

Table 6.1 Compressions of *greasy*

Transform	*Sig. values*	*Bpp*	*RMS Error*
wavelet	3943	0.80	0.760
wavelet packet	3162	0.56	0.817

In Figure 6.1(a) we show the *Barbara* image. If a 4-level Daub 9/7 wavelet transform is applied to this image using 0.5 bpp, then the compressed image is virtually indistinguishable from the original image; so we will not display it. There are some noticeable differences in details at sufficiently high magnification, as can be seen by comparing Figures 6.1(b) and 6.1(c). If instead, we compute a 4-level Daub 9/7 wavelet packet transform and also use 0.5 bpp, then the wavelet packet transform performs significantly better in several respects. As summarized in Table 6.2, wavelet packet compression achieves higher PSNRs with less bits (engineers call this *coding gain*). There is also improved accuracy of detail in the wavelet packet compression, as shown in Figure 6.1(d). In particular, the two sets of diagonal stripes aligned along the two vertical folds of *Barbara*'s scarf are better preserved in Figure 6.1(d) than in Figure 6.1(c).

There is insufficient space to pursue a thorough explanation of why the wavelet packet transform performs better in this example. However, we can gain some understanding by comparing a Daub 9/7 wavelet transform of *Barbara* with a Daub 9/7 wavelet packet transform. In Figure 6.2(a) we show a 2-level Daub 9/7 wavelet transform of the *Barbara* image. The 1-level fluctuations, $\mathbf{h}^1$, $\mathbf{d}^1$, and $\mathbf{v}^1$, all reveal considerable structure. It is easy to discern ghostly versions of the original *Barbara* image within each of these fluctuations. The presence of these ghostly subimages suggests the possibility of

© 2008 by Taylor & Francis Group, LLC

Table 6.2 Compressions of *Barbara*

Transform	*Bpp*	*Sig. values*	*PSNR*
wavelet	0.50	25586	30.84
wavelet packet	0.50	20631	31.50
wavelet	0.25	12415	26.95
wavelet packet	0.25	9936	28.21

performing wavelet transform compression on all of these subimages. In other words, we compute another 1-level wavelet transform on each of these fluctuation subimages, which produces the 2-level wavelet packet transform shown in Figure 6.2(b). Notice that, in the regions corresponding to the horizontal fluctuation $\mathbf{h}^1$ and the diagonal fluctuation $\mathbf{d}^1$ in the wavelet transform, there is a considerable reduction in the number of significant values in the wavelet packet transform. This reduction enables a greater compression of the *Barbara* image. For similar reasons, the 4-level wavelet packet transform exhibits a more compact distribution of significant coefficients, hence a greater

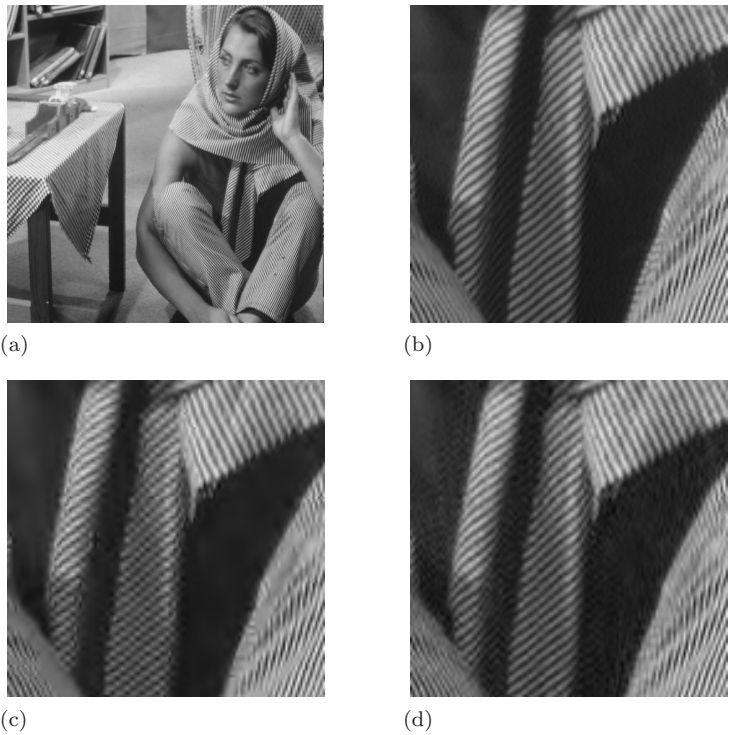
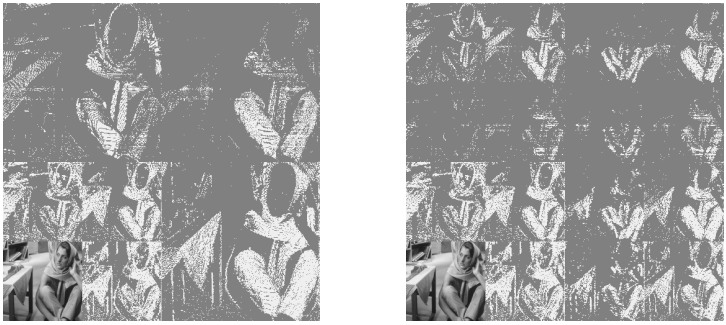


FIGURE 6.1
(a) *Barbara* image. (b) Detail of original. (c) Detail of wavelet compressed image, 0.5 bpp, PSNR = 28.18. (d) Detail of wavelet packet compressed image, 0.5 bpp, PSNR = 29.44.

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 6.2
(a) 2-level Daub 9/7 wavelet transform of *Barbara* image with magnitudes of fluctuation values larger than 8 shown in white. (b) Similar display for the 2-level Daub 9/7 wavelet packet transform of *Barbara* image.

compression, than the 4-level wavelet transform.

6.2.1 Fingerprint compression

This last example also gives us some further insight into the WSQ method of fingerprint compression. WSQ achieves *at least* 15:1 compression, without noticeable loss of detail, on all fingerprint images. It achieves such a remarkable result by applying a wavelet packet transform for which not every subimage is subjected to a further 1-level wavelet transform, but a large percentage of subimages are further transformed. To illustrate why this might be advantageous, consider the two transforms of the *Barbara* image in Figure 6.2. Notice that the vertical fluctuation $\mathbf{v}^1$ in the lower right quadrant of Figure 6.2(a) does not seem to be significantly compressed by applying another 1-level wavelet transform. Therefore, some advantage in compression might be obtained by *not* applying a 1-level wavelet transform to this subimage, while applying it to the other subimages. The wavelet packet transform used by WSQ is described in [6] and [7]. It should also be noted that JPEG 2000 Part 2 allows for wavelet packet transforms, see p. 606 of [7].

6.3 Continuous wavelet transforms

In this section and the next we shall describe the concept of a *continuous wavelet transform* (CWT), and how it can be approximated in a discrete form using a computer. We begin our discussion by describing one type of CWT, known as the *Mexican hat* CWT, which has been used extensively in seismic analysis. In the next section we turn to a second type of CWT, the Gabor CWT, which has many applications to analyzing audio signals. Although we do not have space for a thorough treatment of CWTs, we can nevertheless introduce some of the essential ideas.

© 2008 by Taylor & Francis Group, LLC

The notion of a CWT is founded upon many of the concepts that we introduced in our discussion of discrete wavelet analysis in Chapters 2 through 5, especially the ideas connected with discrete correlations and frequency analysis. A CWT provides a very redundant, but also very finely detailed, description of a signal in terms of both time and frequency. CWTs are particularly helpful in tackling problems involving signal identification and detection of hidden transients (hard to detect, short-lived elements of a signal).

To define a CWT we begin with a given function $\Psi(x)$, which is called the *analyzing wavelet.* For instance, if we define $\Psi(x)$ by

$$\Psi(x) = 2\pi w^{-1/2} \left[1 - 2\pi (x/w)^2 \right] e^{-\pi (x/w)^2}, \quad w = 1/16, \tag{6.7}$$

then this analyzing wavelet is called a *Mexican hat wavelet,* with *width parameter* $w = 1/16$. See Figure 6.3(a).

It is possible to choose other values for w besides $1/16$, but this one example should suffice. By graphing the Mexican hat wavelet using different values of w, it is easy to see why w is called a width parameter. The smaller the value of w, the more the energy of $\Psi(x)$ is confined to a smaller interval of the x-axis.

The Mexican hat wavelet is not the only kind of analyzing wavelet. In the next section, we shall consider the Gabor wavelet, which is very advantageous for analyzing recordings of speech or music. We begin in this section with the Mexican hat wavelet because it allows us to more easily explain the concept of a CWT.

Given an analyzing wavelet $\Psi(x)$, a CWT of a discrete signal $\mathbf{f}$ is defined by computing several correlations of this signal with discrete samplings of the functions $\Psi_s(x)$ defined by

$$\Psi_s(x) = \frac{1}{\sqrt{s}} \Psi \left(\frac{x}{s} \right), \quad s > 0. \tag{6.8}$$

The parameter s is called a *scale parameter.* If we sample each signal $\Psi_s(x)$ at discrete time values $t_1, t_2, \ldots, t_N$, where N is the length of $\mathbf{f}$, then we generate the discrete signals $\mathbf{g}_s$ defined by

$$\mathbf{g}_s = (\Psi_s(t_1), \Psi_s(t_2), \ldots, \Psi_s(t_N)) .$$

A CWT of $\mathbf{f}$ then consists of a collection of discrete correlations $(\mathbf{f} : \mathbf{g}_s)$ over a finite collection of values of s. A common choice for these values is

$$s = 2^{-k/M}, \quad k = 0, 1, 2, \ldots, I \cdot M$$

where the positive integer I is called the number of *octaves* and the positive integer M is called the number of *voices* per octave. For example, 8 octaves and 16 voices per octave is the default choice in FAWAV. Another popular choice is 6 octaves and 12 voices per octave. This latter choice of scales corresponds—based on the relationship between scales and frequencies that

© 2008 by Taylor & Francis Group, LLC

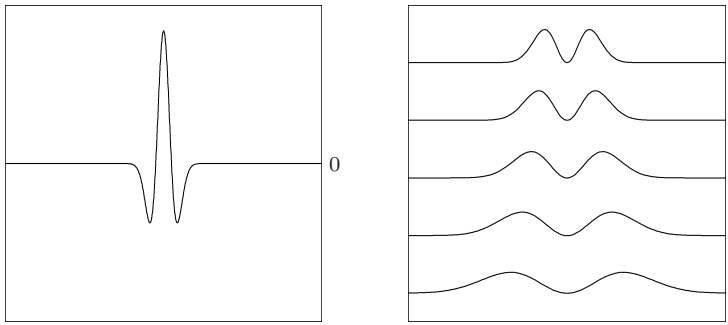


FIGURE 6.3
(a) The Mexican hat wavelet, $w = 1/16$. (b) DFTs of discrete samplings of this wavelet for scales $s = 2^{-k/6}$, from $k = 0$ at the top, then $k = 2$, then $k = 4$, then $k = 6$, down to $k = 8$ at the bottom.

we describe below—to the scale of notes on a piano (also known as the *well-tempered scale*).

The purpose of computing all these correlations is that *a very finely detailed time-frequency analysis* of a signal can be carried out by making a judicious choice of the width parameter w and the number of octaves and voices. To see this, we observe that Formula (5.40) tells us that the DFTs of the correlations $(\mathbf{f} : \mathbf{g}_s)$ satisfy

$$(\mathbf{f} : \mathbf{g}_s) \stackrel{\mathcal{F}}{\longmapsto} \mathcal{F}\mathbf{f}\,\overline{\mathcal{F}\mathbf{g}_s}. \tag{6.9}$$

For a Mexican hat wavelet, $\mathcal{F}\mathbf{g}_s$ is real-valued; hence $\overline{\mathcal{F}\mathbf{g}_s} = \mathcal{F}\mathbf{g}_s$. Therefore Equation (6.9) becomes

$$(\mathbf{f} : \mathbf{g}_s) \stackrel{\mathcal{F}}{\longmapsto} \mathcal{F}\mathbf{f}\,\mathcal{F}\mathbf{g}_s. \tag{6.10}$$

Formula (6.10) is the basis for a very finely detailed time-frequency decomposition of a discrete signal $\mathbf{f}$. For example, in Figure 6.3(b) we show graphs of the DFTs $\mathcal{F}\mathbf{g}_s$ for the scale values $s = 2^{-k/6}$, with $k = 0$, 2, 4, 6, and 8. These graphs show that when these DFTs are multiplied with the DFT of $\mathbf{f}$, they provide a decomposition of $\mathcal{F}\mathbf{f}$ into a succession of frequency bands. These successive bands overlap each other, thus providing a very redundant decomposition of the DFT of $\mathbf{f}$. Notice that the bands containing higher frequencies correspond to smaller scale values; *there is a reciprocal relationship between scale values and frequency values.* It is also important to note that there is a decomposition in time, due to the essentially finite width of the Mexican hat wavelet [see Figure 6.3(a)].

A couple of examples should help to clarify these points. The first example we shall consider is a test case designed to illustrate the connection between a CWT and the time-location of frequencies in a signal. The second example is an illustration of how a CWT can be used for analyzing an ECG signal.

© 2008 by Taylor & Francis Group, LLC

For our first example, we shall analyze a discrete signal $\mathbf{f}$, obtained from 2048 equally spaced samples of the following analog signal:

$$\begin{aligned} &\sin(40\pi x)e^{-100\pi(x-.2)^2} \\ &+ \left[\sin(40\pi x) + 2\cos(160\pi x)\right] e^{-50\pi(x-.5)^2} \\ &+ 2\sin(160\pi x)e^{-100\pi(x-.8)^2} \end{aligned} \tag{6.11}$$

over the interval $0 \le x \le 1$. See the top of Figure 6.4(a).

The signal in (6.11) consists of three terms. The first term contains a sine factor, $\sin(40\pi x)$, of frequency 20. Its other factor, $e^{-100\pi(x-.2)^2}$, serves as a damping factor which limits the energy of this term to a small interval centered on $x = 0.2$. This first term appears most prominently on the left-third of the graph at the top of Figure 6.4(a). Likewise, the third term contains a sine factor, $2\sin(160\pi x)$, of frequency 80, and this term appears most prominently on the right-third of the signal's graph. Notice that this frequency of 80 is four times as large as the first frequency of 20. Finally, the middle term

$$\left[\sin(40\pi x) + 2\cos(160\pi x)\right] e^{-50\pi(x-.5)^2}$$

has a factor containing both of these two frequencies, and can be observed most prominently within the middle of the signal's graph.

The CWT, also known as a *scalogram,* for this signal is shown at the bottom of Figure 6.4(a). The analyzing wavelet used to produce this CWT was a Mexican hat wavelet of width 1/16, with scales ranging over 8 octaves and 16 voices. The labels on the right side of the figure indicate *reciprocals* of the scales used. Because of the reciprocal relationship between scale and frequency noted above, this reciprocal-scale axis can also be viewed as a frequency axis. Notice that the four most prominent portions of this scalogram are aligned directly below the three most prominent parts of the signal. Of equal importance is the fact that these four portions of the scalogram are centered on two reciprocal-scales, $1/s \approx 2^{2.2}$ and $1/s \approx 2^{4.2}$. The second reciprocal scale is four times larger than the first reciprocal scale, just as the frequency 80 is four times larger than the frequency 20. Bearing this fact in mind, and recalling the alignment of the prominent regions of the scalogram with the three parts of the signal, we can see that the CWT provides us with a *time-frequency portrait* of the signal.

We have shown that it is possible to correctly interpret the meaning of this scalogram; nevertheless, we can produce a much simpler and more easily interpretable scalogram for this test signal using a Gabor analyzing wavelet. See Figure 6.5(a). We shall discuss this Gabor scalogram in the next section.

Our second example uses a Mexican hat CWT for analyzing a signal containing several transient bursts, a simulated ECG signal first considered in Section 5.4. See the top of Figure 6.4(b). The bottom of Figure 6.4(b) is a scalogram of this signal using a Mexican hat wavelet of width 2, over a range of 8 octaves and 16 voices. This scalogram shows how a Mexican hat wavelet

© 2008 by Taylor & Francis Group, LLC

can be used for detecting the onset and demise of each heartbeat. In particular, the aberrant, fourth heartbeat is singled out from the others by the longer vertical ridges extending upwards to the highest frequencies (at the eighth octave). Although this example is only a simulation, it does show the ease with which the Mexican hat CWT detects the presence of short-lived parts of a signal. Similar identifications of transient bursts are needed in seismology for the detection of earthquake tremors. Consequently, Mexican hat wavelets are widely used in seismology.

6.4 Gabor wavelets and speech analysis

In this section we describe Gabor wavelets, which are similar to the Mexican hat wavelets examined in the previous section, but provide a more powerful tool for analyzing speech and music. We shall first go over their definition, and then illustrate their use with some examples.

A *Gabor wavelet,* with width parameter w and frequency parameter ν, is the following analyzing wavelet:

$$\Psi(x) = w^{-1/2} e^{-\pi(x/w)^2} e^{i2\pi\nu x/w}. \tag{6.12}$$

This wavelet is complex valued. Its real part $\Psi_{\mathrm{R}}(x)$ and imaginary part $\Psi_{\mathrm{I}}(x)$ are

$$\Psi_{\mathrm{R}}(x) = w^{-1/2} e^{-\pi(x/w)^2} \cos(2\pi\nu x/w), \tag{6.13a}$$

$$\Psi_{\mathrm{I}}(x) = w^{-1/2} e^{-\pi(x/w)^2} \sin(2\pi\nu x/w). \tag{6.13b}$$

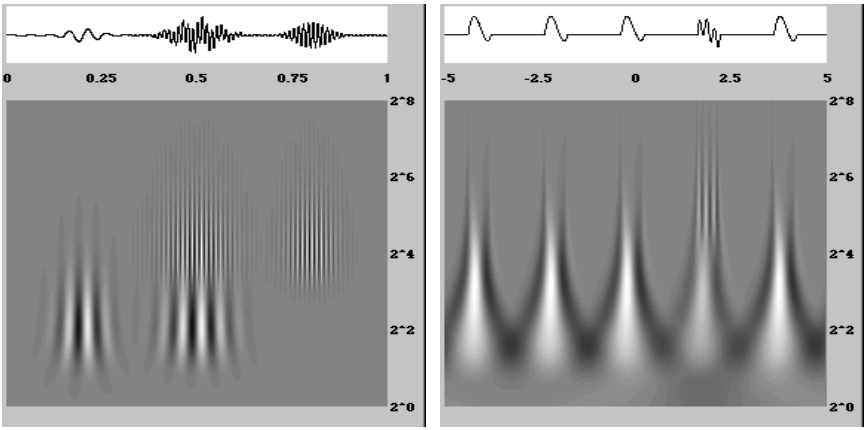
The width parameter w plays the same role as for the Mexican hat wavelet; it controls the width of the region over which most of the energy of $\Psi(x)$ is concentrated. The value ν/w is called the *base frequency* for a Gabor CWT.

One advantage that Gabor wavelets have when analyzing sound signals is that they contain factors of cosines and sines, as shown in (6.13a) and (6.13b). These cosine and sine factors allow the Gabor wavelets to create easily interpretable scalograms of those signals which are combinations of cosines and sines—the most common instances of such signals are recorded music and speech. We shall see this in a moment, but first we need to say a little more about the CWT defined by a Gabor analyzing wavelet.

Because a Gabor wavelet is complex valued, it produces a complex-valued CWT. For many signals, it is often sufficient to just examine the magnitudes of the Gabor CWT values. In particular, this is the case with the signals analyzed in the following examples.

For our first example, we use a Gabor wavelet with width 1 and frequency 5 for analyzing the signal in (6.11). The graph of this signal is shown at the top of Figure 6.5(a). As we discussed in the previous section, this signal consists of three portions with associated frequencies of 20 and 80. The magnitudes for a Gabor scalogram of this signal, using 8 octaves and 16 voices, are graphed

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 6.4
(a) Mexican hat CWT (scalogram) of a test signal with two main frequencies. (b) Mexican hat scalogram of simulated ECG signal. Whiter colors represent positive values, blacker values represent negative values, and the gray background represents zero values.

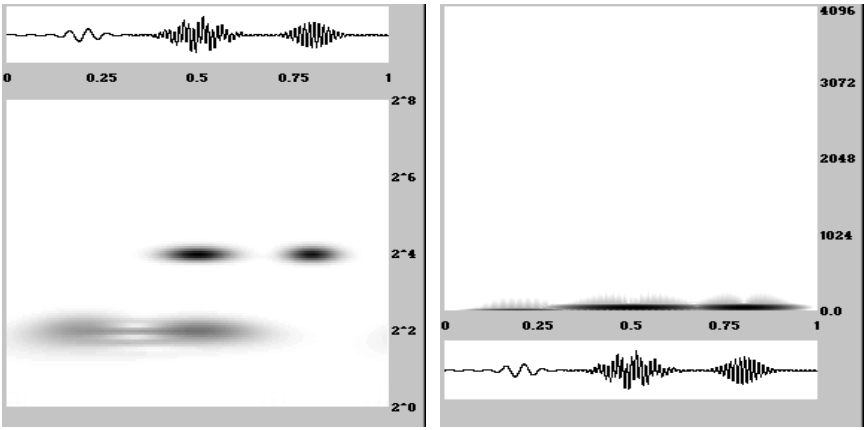
at the bottom of Figure 6.5(a). We see that this *magnitude-scalogram* consists of essentially just four prominent spots aligned directly below the three most prominent portions of the signal. These four spots are centered on the two reciprocal-scale values of 2^2 and 2^4, which are in the same ratio as the two frequencies 20 and 80. [Notice also that the base frequency is $5/1 = 5$ and that $20 = 5 \cdot 2^2$, $80 = 5 \cdot 2^4$.] It is interesting to compare this scalogram with a spectrogram of the test signal shown in Figure 6.5(b). The spectrogram has all of its significant values crowded together at the lower frequencies; the scalogram is zooming in on this lower range of frequencies, from 5 Hz to 1280 Hz along an octave-based scale.

It is interesting to compare Figures 6.4(a) and 6.5(a). The simplicity of Figure 6.5(a) makes it much easier to interpret. The reason that the Gabor CWT is so clean and simple is because, for the proper choices of width w and frequency ν, the test signal in (6.11) consists of terms that are identical in form to one of the functions in (6.13a) or (6.13b). Therefore, when a scale value s produces a function $\Phi_{\mathrm{R}}(x/s)/\sqrt{s}$, or a function $\Phi_{\mathrm{I}}(x/s)/\sqrt{s}$, having a form similar to one of the terms in (6.11), then the correlation $(\mathbf{f} : \mathbf{g}_s)$ in the CWT will have some high-magnitude values.

This first example might appear to be rather limited in scope. After all, how many signals encountered in the real world are so nicely put together as this test signal? Our next example, however, shows that a Gabor CWT performs equally well in analyzing a real signal: a speech signal.

In Figure 6.6(b) we show a Gabor magnitude-scalogram of a recording of the author saying the word *call*. The recorded signal, which is shown at the top of the figure, consists of three main portions. The first two portions

© 2008 by Taylor & Francis Group, LLC



(a) (b)

FIGURE 6.5
(a) Magnitudes of Gabor scalogram of test signal. (b) Spectrogram of test signal. Darker regions denote larger magnitudes; lighter regions denote smaller magnitudes.

correspond to the two sounds, *ca* and *ll*, that form the word *call*. The *ca* portion occupies a narrow area on the far left side of the *call* signal's graph, while the *ll* portion occupies a much larger area consisting of the middle half of the *call* signal's graph. The third portion lies at the right end of the signal and is a "clipping sound" that is the start of the consonant "b" that begins the word "back" (the *call* signal was clipped from a recording of the author speaking the phrase "you call back"). For comparison, we have also plotted the *call* signal's spectrogram in Figure 6.6(a). Again, as with our test signal, the scalogram is able to zoom in and better display the time-frequency structure of this speech signal.

To analyze the *call* signal, we used a Gabor wavelet of width 1/8 and frequency 10 (hence a base frequency of $10/(1/8) = 80$ Hz), with scales ranging over 4 octaves and 16 voices. The resulting magnitude-scalogram is composed of several regions. The largest region is a collection of several horizontal bands lying below the *ll* portion. We shall concentrate on analyzing this region.

This region below the *ll* portion consists of seven horizontal bands centered on the following approximate reciprocal-scale values:

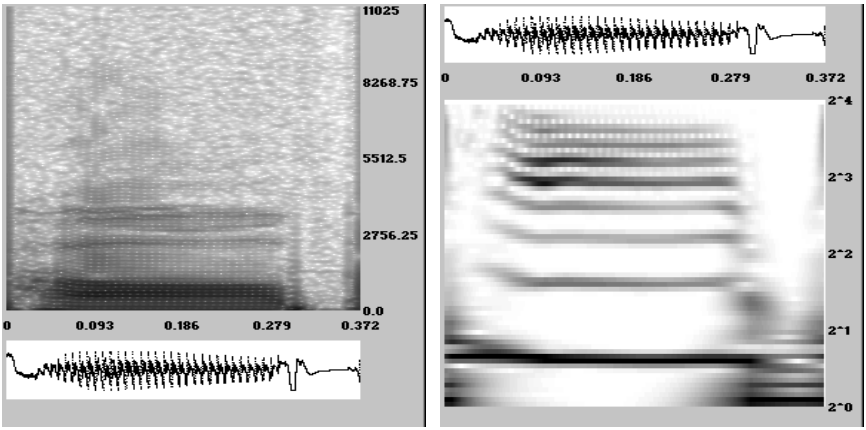
$$2^{0.625},\ 2^{1.625},\ 2^{2.188},\ 2^{2.625},\ 2^{2.953},\ 2^{2.97},\ 2^{3.219},\ 2^{3.625}. \tag{6.14}$$

If we divide each of these values by the smallest one, $2^{0.625}$, we get the following approximate ratios:

$$1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8. \tag{6.15}$$

Since reciprocal-scale values correspond to frequencies, we can see that these bands correspond to frequencies on a harmonic (musical) scale. In fact, in Figure 6.7(b) we show a graph of the spectrum of a sound clip of the *ll*

© 2008 by Taylor & Francis Group, LLC



(a) (b)

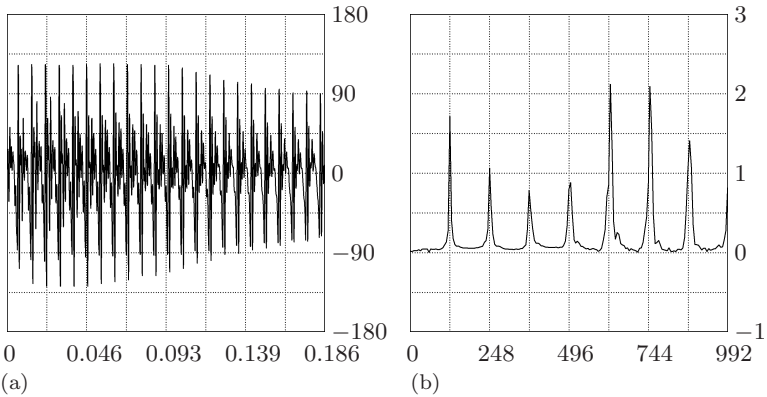
FIGURE 6.6
(a) Spectrogram of *call* sound. (b) Magnitudes of its Gabor scalogram.

portion of the *call* signal. This spectrum shows that the frequencies of peak energy in the *ll* portion have the following approximate values:

$$124,\ 248,\ 372,\ 496,\ 620,\ 744,\ 868,\ 992. \tag{6.16}$$

Notice that these frequencies have the same ratios to the lowest frequency of 124 as the ratios in (6.15) [and that $124 \approx 80 \cdot 2^{0.625}$, ..., $992 \approx 80 \cdot 2^{3.625}$].

This region of the scalogram illustrates an important property of many portions of speech signals, the property of *frequency banding.* These frequency bands are called *formants* in linguistics. All speakers, whether they are native speakers of English or not, produce a sequence of such frequency bands for the *ll* portion of *call.* For some speakers, the bands are horizontal, while for



(a) (b)

FIGURE 6.7
(a) A portion of the *ll* sound in the *call* signal; the horizontal axis is the time axis. (b) Spectrum of signal in (a); the horizontal axis is the frequency axis.

© 2008 by Taylor & Francis Group, LLC

other speakers the bands are curved. The *ll* sound is a fundamental unit of English speech, called a *phoneme.*

Notice that there are also curved formants directly preceding the straight formants of the *ll* sound; these formants describe the phoneme *aa* that immediately precedes the phoneme for *ll* in the word *call.* There are also two other structures in the scalogram that do not exhibit formants; these occur at the beginning and end of the recording. At the beginning there is a hard *c* sound corresponding to the beginning of the word *call.* In the scalogram we see that the portion at the far left is composed of a widely dispersed, almost continuous, range of frequencies without significant banding—this reflects the percussive nature of the hard consonant phoneme *c.* There is a similar continuous range of frequencies at the end of the *call* signal corresponding to the "clipping" sound that begins the consonant *b.*

This example shows what a powerful tool the Gabor CWT provides for analyzing a speech signal. We have used it to clearly distinguish the phonemes in the *call* sound, to understand the formant structure of its *ll* portion, and to determine that its consonants lack a formant structure.

Another application of these Gabor scalograms is that, when applied to recordings of different people saying *call*, they produce visibly different scalograms. These scalograms function as "fingerprints" for identifying different speakers. The ribbon structure of formants for the *ll* portion is displayed for all speakers, although tracing out different curves for different speakers.

6.4.1 Musical analysis: formants in song lyrics

We close this section by showing how Gabor scalograms, in concert with spectrograms, provide a powerful time-frequency approach to analyzing music. We already illustrated this once in Figure 5.15 (see p. 199). The image in Figure 5.15(b) is a Gabor scalogram that zooms in on the spectrogram in Figure 5.15(a) over a two-octave frequency range of 500 to 2000 Hz.

Another interesting musical example is shown in Figure 6.8. The image in Figure 6.8(a) is a Blackman windowed spectrogram of a passage from the song *Buenos Aires.* The dark curved ribbons are the formants of the singer's voice as she sings the lyrics "you'll be on me too." In Figure 6.8(b) we show a scalogram that zooms in on a 3-octave range of the spectrogram spanning 200 to 1600 Hz.[2] The time-frequency analysis shown in these two images reveals aspects of the Multiresolution Principle at work: there are evident repetitions of time-frequency structures and substructures at different locations and with different sizes.

It is interesting to compare Figures 5.15 and 6.8. The former is based on an instrumental Chinese folk melody and the latter is based on a modern, popular English song lyric. Notice that there is some similarity between the small angular structures: in the latter case, they are formants of the singer's voice; in the former case, they are "pitch excursions" of a Chinese stringed

[2]See Section 6.7 for more details on how we created Figure 6.8(b).

© 2008 by Taylor & Francis Group, LLC

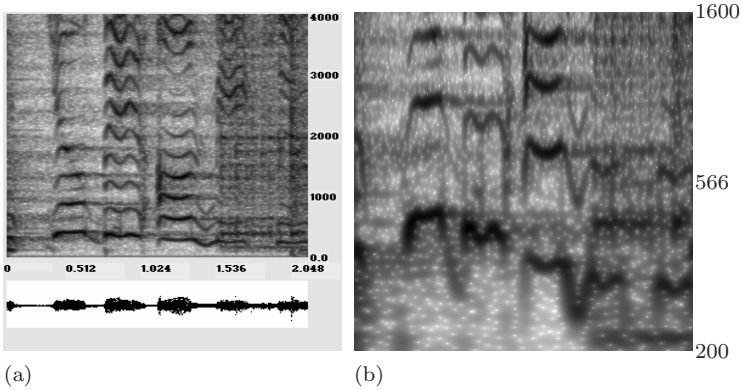


FIGURE 6.8
Time-frequency analysis of a passage from the song *Buenos Aires.* (a) Spectrogram. (b) Zooming in on three octaves of the frequency range of (a).

instrument that extends the power and range of human voice.

6.5 Percussion scalograms and musical rhythm

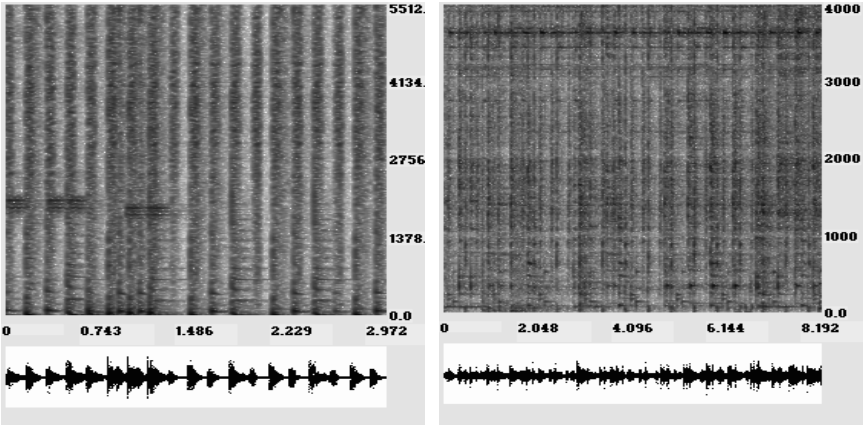
For our final discussion of the primer, we describe how time-frequency methods can be used to analyze musical rhythm. In particular, we show how a new technique known as *percussion scalograms* makes use of both spectrograms and scalograms to analyze the multiple time-scales occurring within the rhythms of a percussion performance.

Our discussion will focus on two percussion sequences. The first sequence is an introductory passage from the song, *El Matador,* which is saved as the file `el_matador_percussion_clip.wav` at the primer website. Listening to this file you will hear a relatively simple rhythm of drum beats, with one short shift in tempo, and with several whistle blowings as accompaniment. We shall use this passage to illustrate the basic principles underlying our approach. The second sequence, which will show the power of our method, is a complex Latin percussion passage that introduces the song *Buenos Aires.* This passage is saved in the file `Buenos Aires percussion clip.wav` at the primer website. Listening to this file you will hear a richly structured percussion performance with several tempo shifts (as well as some high pitch background sound). Our percussion scalogram method will produce an objective description of these tempo shifts that accords well with our aural perception.

To derive our percussion scalogram method for analyzing drum rhythms, we consider the percussion sequence from the beginning of *El Matador.* In Figure 6.9(a) we show its spectrogram. This spectrogram is mostly composed of a sequence of thick vertical segments, which we will call *vertical swatches.*[3] Each vertical swatch corresponds to a percussive strike on a drum. These

[3]The whistle blowings correspond to three rectangular blotches at the left center of the spectrogram.

© 2008 by Taylor & Francis Group, LLC



(a) Passage from *El Matador* (b) Passage from *Buenos Aires*

FIGURE 6.9
Spectrograms from two percussion sequences.

sharp strikes on drum heads excite a continuum of frequencies rather than a discrete tonal sequence of fundamentals and overtones. The rapid onset and decay of these strike sounds produces vertical swatches in the time-frequency plane. A more complex pattern of thinner vertical swatches can be seen in the spectrogram of the *Buenos Aires* percussion passage in Figure 6.9(b).

Our percussion scalogram method has the following two parts:

I. *Pulse train generation.* We generate a "pulse train," a sequence of alternating intervals of 1-values and 0-values (see the bottom graph in Figure 6.10). The location and duration of the intervals of 1-values corresponds to our hearing of the drum strikes, and the location and duration of the intervals of 0-values corresponds to the silences between the strikes. In Figure 6.10, the rectangular-shaped pulses correspond to sharp onset and decay of transient bursts in the percussion signal graphed just above the pulse train. The widths of these pulses are approximately equal to the widths of the vertical swatches shown in the spectrogram (we graphed only a portion of the spectrogram that omits the blotches from the whistle blowings, so as to isolate just the drum strikings). In Steps 1 and 2 of the method below we describe how this pulse train is generated.

II. *Gabor CWT.* We use a Gabor CWT to analyze the pulse train. The rationale for doing this is that the pulse train is a step function analog of a sinusoidal of varying frequency. Because of this rough correlation between tempo of pulses in a pulse train and frequency in sinusoidal curves, we employ a Gabor CWT for analysis.[4] For example, see Figure 6.11. The thick vertical line segments in the top half of the scalogram correspond

[4]In his thesis [18], Leigh M. Smith provides a thorough empirical study of the efficacy of using Gabor CWTs to analyze percussive pulse trains.

© 2008 by Taylor & Francis Group, LLC

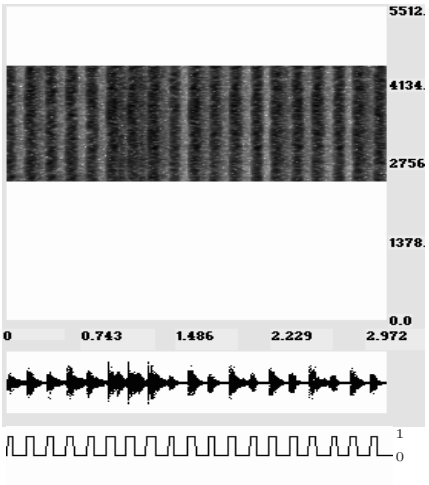


FIGURE 6.10
Pulse Train for the *El Matador* percussion sequence.

to the drum strikes, and there is a connecting region at the bottom of three of the segments (the 6^{th}, 7^{th}, and 8^{th} segments counting from the left). When listening to the passage we hear those three strikes as a group with a clearly defined tempo shift.[5] This CWT calculation is performed in Step 3 of the method.

Now that we have outlined the basis for the percussion scalogram method, we can list it in detail. The percussion scalogram method for analyzing percussive rhythm consists of the following three steps.

Percussion Scalogram Method

Step 1. *Compute a signal consisting of averages of the Gabor transform square-magnitudes for horizontal slices lying within a frequency range that consists mostly of vertical swatches.* For the time intervals corresponding to vertical swatches in the spectrogram (for example, as shown in Figure 6.10) this step will produce higher square-magnitude values that lie above the mean of all square-magnitudes (because the mean is pulled down by the intervals of silence). For the *El Matador* sequence, the frequency range of 2500 to 4500 Hz was used, as it consists mostly of the vertical swatches corresponding to the percussive strikes. (Note: In a *purely percussive passage,* containing only drum strikes without any background sounds, the complete frequency range can be used.)

[5] An important feature of the FAWAV program used to generate this percussion scalogram is that the sound file can be played and a cursor will travel across the percussion scalogram, confirming our statements about the meaning of its features.

© 2008 by Taylor & Francis Group, LLC

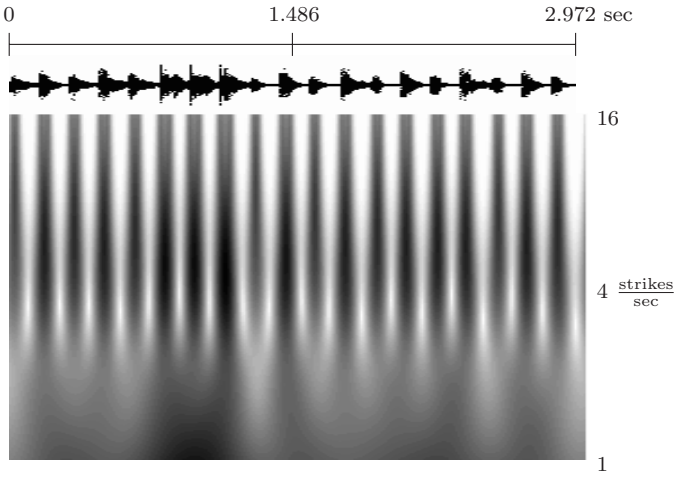


FIGURE 6.11
Rhythmic analysis of *El Matador* percussion sequence. The percussion sequence is graphed on top, and below it is its percussion scalogram using 4 octaves, 64 voices, width 0.5, freq. 0.5 (obtained from the frequency range 2500 to 4500 Hz of its spectrogram).

Step 2. *Compute a signal that is* 1 *whenever the signal from Step 1 is larger than its mean and* 0 *otherwise.* As the discussion in Step 1 shows, this will produce a pulse train whose intervals of 1-values mark off the position and duration of the vertical swatches (hence of the drum strikes). Figure 6.10 illustrates this clearly.

Step 3. *Compute a Gabor CWT of the pulse train signal from Step 2.* As we shall now discuss, this Gabor CWT provides an objective picture of the varying rhythms within a percussion performance.

We have already discussed the Gabor CWT shown in Figure 6.11 for the *El Matador* percussion sequence. The only points we want to add to that discussion are some details on the meanings of the parameters used for the Gabor CWT. To create the Gabor CWT in Figure 6.11, we used a width parameter of 0.5 and a frequency parameter of 0.5. That yields a base frequency of 1.0 strikes/sec. The range of 4 octaves that we used then gives an upper frequency value of 16 strikes/sec.[6] Notice that the vertical bars in the scalogram in Figure 6.11 are centered on a frequency value of about 6 or 7 strikes/sec and that corresponds to the number of strikes that one detects within any given 1 second interval; this illustrates that it is correct to interpret the vertical axis for the scalogram as an (octave-scaled) frequency axis of strikes/sec.

[6]These CWT parameter values were obtained empirically. An a priori (automatic) selection method is the subject of current research. See the preprint [22].

© 2008 by Taylor & Francis Group, LLC

6.5.1 Analysis of a complex percussive rhythm

As an illustration of the power of our method, we use a percussion scalogram to analyze the complex rhythms of the opening percussion passage from the *Buenos Aires* song. See Figure 6.12. To isolate the percussive sounds, the drum strikes, from the rest of the sounds in the passage, we used a frequency range of 2000 to 3000 Hz in Step 1 of the percussion scalogram method. The parameters of the Gabor CWT are specified in the caption of Figure 6.12.

As with the *El Matador* sequence, it helps to play the recording of the percussion passage and watch the cursor trace over the percussion scalogram. After listening a couple of times, and watching the cursor run along the top of the scalogram, you should find that the thin vertical strips at the top of the scalogram correspond to the individual drum strikes. What is even more interesting, however, is that several of these vertical strips bind together into larger blobs lower down on the frequency scale (for instance, the blobs above the labels γ_1 to γ_5 in the figure). If you listen again to the recording and watch the cursor as it passes these blobs γ_1 to γ_5, you will notice that the strikes occur in groups that correspond precisely to these blobs. Furthermore, the blobs γ_3 to γ_5 are connected together, and one does perceive a larger time-scale grouping of percussion strikes over the time-interval covered by these three blobs. Finally, we note that there is another collection of blobs to the right of γ_5. We have not labeled them, but we leave it to the reader to infer the connection between them and the shifting pattern of drum strikes in the recording. Notice, however, that these blobs appear to be linked to a larger region, labeled $\mathbf{\Gamma}$, which provides an objective description of our aural perception of the further grouping of these collections of drum strikes.

6.5.2 Multiresolution Principle for rhythm

Our discussion of these two percussion sequences illustrates the fact that the Multiresolution Principle for tonal music that we introduced on p. 196—the patterning of time-frequency structures over multiple time-scales—also applies to rhythmic percussion. We can even see the three representations described by Pinker (p. 200) applying as well, if we substitute "strikes" for "notes." For example, the single strikes are grouped into blobs, and some of these blobs are joined together into longer groups. This multiresolution time-frequency patterning, captured by our percussion scalograms, may be useful in characterizing different styles of percussion. But that is a subject for future research.

6.6 Notes and references

The best introductory material on wavelet packet transforms can be found in [2] and [3]. There is also a good discussion in [4]. A very thorough treatment of the subject is given in [5]. The relation between wavelet packet transforms and the WSQ method is described in [6], and the wavelet packet transform

© 2008 by Taylor & Francis Group, LLC

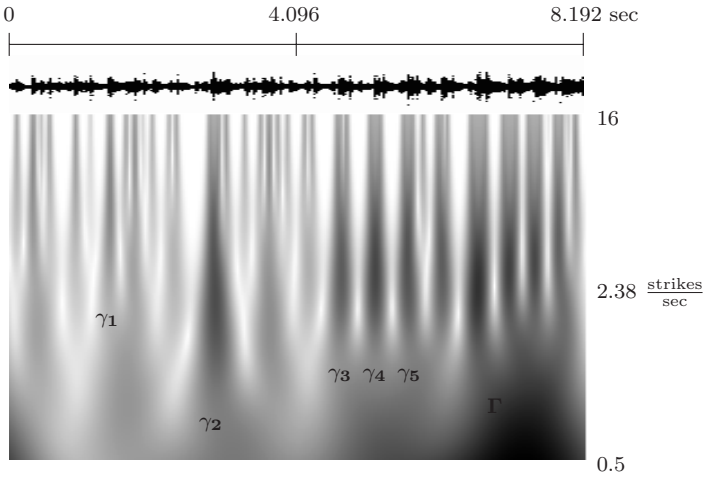


FIGURE 6.12
Rhythmic analysis of *Buenos Aires* percussion sequence. The percussion sequence is graphed on top, and below it is its percussion scalogram using 5 octaves, 51 voices, width 2, freq. 1 (obtained from the frequency range 2000 to 3000 Hz of its spectrogram). The labels are explained in the text.

option allowed by JPEG 2000 Part 2 is described in [7].

Rigorous expositions of the complete theory of CWTs can be found in [8] and [9]. A more complete treatment of the discrete version described in this primer is given in [10].

For a discussion of the uses of the CWT for analysis of ECGs, see [11] and [12]. In addition to an excellent discussion of the topic, [12] also contains an extensive bibliography.

Applying Gabor CWTs to the detection of engine malfunctions in Japanese automobiles is described in [13]. An interesting relationship between CWTs and human hearing, with applications to speech analysis, is described in [14]. Background on formants and phonemes in linguistics can be found in [15].

Scalograms are used, in conjunction with spectrograms, to provide analysis of music and musical instruments in [16] and [17]. An empirical discussion of percussive rhythm and applications is given by Leigh Smith in [18]. Some other papers on related topics can be found on his webpage [19]. William Sethares has done profound work on computerized rhythm analysis [20], [21]. The preprint [22] describes an alternative approach to this same topic.

6.6.1 Additional references

We now provide references for some additional topics that extend the discussion in the text. These topics are (1) best basis algorithms, (2) local cosine series, (3) frames, (4) curvelets, (5) 3D wavelets, and (6) video compression.

Best basis algorithms. There are algorithms that choose which fluctuation subsignals to decompose further (by wavelet transform) according to some

© 2008 by Taylor & Francis Group, LLC

minimization of a *cost function.* An excellent introduction to such a best basis algorithm, and its application to the WSQ algorithm, can be found in David Walnut's book [23]. A complete discussion, by the discoverer of the technique, is in [5].

Local cosine series. The Gabor transform discussed in Chapter 5 has been criticized for its reliance on windows of constant size. A method which allows for windows of varying size, as well as using real-valued cosine functions, is the method of *local cosine series.* (The method of choosing windows of varying size can be adapted to the multi-window Gabor transforms described in problem 5.9.6 on p. 218. Further discussion of multi-window Gabor transforms can be found in [24].) Local cosine series have proven to be quite useful in compression of signals. An elementary discussion can be found in Chapter 2 of [25]. See also the paper by Auscher et al. in [26], and the treatment in [5]. A related area to local cosine series are the fields of *lapped orthogonal transforms* and *generalized lapped orthogonal transforms* (GenLOT). These fields are described in [27] to [31].

Frames. The inversion of Gabor transforms discussed in Chapter 5 provides one class of expansions of signals using *frames.* Frames have proven to be especially useful in denoising applications. See [32] to [36]. References [37] and [38] provide fundamental mathematical background.

Curvelets. David Donoho and his collaborators have done extensive work on a generalization of wavelets for image processing known as curvelets. Curvelets provide for efficient modelling of edges in images. See the papers [39] to [41], and also the website [42].

3D wavelets. Wim Sweldens and his collaborators have done a lot of work on wavelets and their generalizations in 3D. See [43] and [44], and the website [45]. Curvelets have also been adapted for 3D processing [46].

Video compression. A good synopsis of the basics of wavelet-based video compression can be found in [10]. Important work in the field is being done by Truong Nguyen's group; see [47] for many downloadable publications.

1. B. Burke. (1994). The mathematical microscope: waves, wavelets, and beyond. In *A Positron Named Priscilla, Scientific Discovery at the Frontier,* M. Bartusiak (Ed.), 196–235, National Academy Press.

2. M.V. Wickerhauser. (1993). Best-adapted wavelet packet bases. In *Different Perspectives on Wavelets,* I. Daubechies (Ed.), AMS, Providence, RI, 155–172.

3. R.R. Coifman and M.V. Wickerhauser. (1993). Wavelets and adapted waveform analysis. A toolkit for signal processing and numerical analysis. In *Different Perspectives on Wavelets,* I. Daubechies (Ed.), AMS, Providence, RI, 119–154.

4. R.R. Coifman and M.V. Wickerhauser. (1994). Wavelets and adapted waveform analysis. In *Wavelets. Mathematics and Applications,* J. Benedetto and M. Frazier (Eds.), CRC Press, Boca Raton, FL, 399–424.

© 2008 by Taylor & Francis Group, LLC

5. M.V. Wickerhauser. (1994). *Adapted Wavelet Analysis from Theory to Software.* A.K. Peters, Wellesley, MA, 1994.

6. J.N. Bradley, C.M. Brislawn, and T. Hopper. (1993). The FBI Wavelet/Scalar Quantization Standard for gray-scale fingerprint image compression. *SPIE,* Vol. 1961, *Visual Information Processing II (1993)*, 293–304.

7. D.S. Taubman and M.W. Marcellin. (2002). *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Kluwer, Boston, MA.

8. I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

9. A.K. Louis, P. Maaß, and A. Rieder. (1997). *Wavelets, Theory and Applications.* Wiley, New York, NY.

10. S. Mallat. (1999). *A Wavelet Tour of Signal Processing. Second Edition.* Academic Press, New York, NY.

11. L. Senhadji, L. Thoraval, and G. Carrault. (1996). Continuous wavelet transform: ECG recognition based on phase and modulus representations and hidden Markov models. In *Wavelets in Medicine and Biology,* A. Aldroubi and M. Unser (Eds.), CRC Press, Boca Raton, FL, 439–464.

12. P.S. Addison. (2005). Wavelet transforms and the ECG: a review. *Physiol. Meas.,* Vol. 26, R155–R199.

13. M. Kobayashi. (1996). Listening for defects: wavelet-based acoustical signal processing in Japan. *SIAM News,* Vol. 29, No. 2.

14. I. Daubechies and S. Maes. (1996). A nonlinear squeezing of the continuous wavelet transform based on auditory nerve models. In *Wavelets in Medicine and Biology,* A. Aldroubi and M. Unser (Eds.), CRC Press, Boca Raton, FL, 527–546.

15. W. O'Grady, M. Dobrovolsky, and M. Arnoff. (1993). *Contemporary Linguistics, An Introduction.* St. Martins Press, New York.

16. J.S. Walker and G.W. Don. (2006). Music: a time-frequency approach. Preprint. Available at http://www.uwec.edu/walkerjs/media/TFAM.pdf

17. J.F. Alm and J.S. Walker. (2002). Time-frequency analysis of musical instruments. *SIAM Review,* Vol. 44, 457–476.

18. L.M. Smith (2000). A multiresolution time-frequency analysis and interpretation of musical rhythm. Thesis, University of Western Australia.

19. L.M. Smith's webpage: http://staff.science.uva.nl/~lsmith/

20. W. Sethares. (2007). *Rhythm and Transforms.* Springer, New York, NY.

21. W. Sethares. (2007). Rhythm and transforms. An extended abstract of his plenary address to the Mathematics and Computation in Music conference in Berlin, May 18, 2007. Available at

http://www.mcm2007.info/pdf/fri1-sethares.pdf

22. X. Cheng, J.V. Hart, and J.S. Walker. (2007). Time-frequency analysis of musical rhythm. Preprint. Available at

http://www.uwec.edu/walkerjs/media/TFAMR.pdf

© 2008 by Taylor & Francis Group, LLC

23. D.F. Walnut. (2002). *An Introduction to Wavelet Analysis.* Birkhäuser, Boston, MA.

24. M. Dörfler and H. Feichtinger. (2004). Quilted Gabor families I: reduced multi-Gabor frames. *Appl. Comput. Harmon. Anal.,* Vol. 356, 2001–2023. Available at

`http://www.mat.univie.ac.at/~moni/`

25. E. Hernandez and G. Weiss. (1996). *A First Course on Wavelets.* CRC Press, Boca Raton, FL.

26. C.K. Chui (Ed.) (1992). *Wavelets: A Tutorial in Theory and Applications.* Academic Press, Boston, MA.

27. H.S. Malvar and D.H. Staelin. (1989). The LOT: transform coding without blocking effects. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. 37, 553–559.

28. T.Q. Nguyen. (1992). A class of generalized cosine-modulated filter bank. *Proceedings 1992 IEEE International Symposium on Circuits and Systems,* Vol. 2, 943–946.

29. R.L. de Queiroz, T.Q. Nguyen, and K.R. Rao. (1996). The GenLOT: generalized linear-phase lapped orthogonal transform. *IEEE Transactions on Signal Processing,* Vol. 44, 497–507.

30. S. Oraintara, P. Heller, T. Tran, and T. Nguyen. (2001). Lattice structure for regular paraunitary linear-phase filterbanks and M-band orthogonal symmetric wavelets. *IEEE Transactions on Signal Processing,* Vol. 49, 2659-2672.

31. Y.-J. Chen, S. Oraintara, and K. Amaratunga. (2005). Dyadic-based factorization for regular paraunitary filter banks and M-band orthogonal wavelets with structural vanishing moments. *IEEE Transactions on Signal Processing,* Vol. 53, 193–207.

32. R. Coifman and Y. Zeevi, Eds. (1988). *Signal and Image Representation in Combined Spaces.* Wavelet Analysis and Applications, Vol. 7, Academic Press, Boston, MA.

33. I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

34. H. Feichtinger and T. Strohmer, Eds. (1998). *Gabor Analysis and Algorithms.* Birkhäuser, Boston, MA.

35. H. Feichtinger and T. Strohmer, Eds. (2002). *Advances in Gabor Analysis.* Birkhäuser, Boston, MA.

36. R. Young. (1980). *An Introduction to Nonharmonic Fourier Series.* Academic Press, New York, NY.

37. P.G. Casazza. (2000). The art of frame theory. *Taiwanese J. of Mathematics,* Vol. 4, 129–201.

38. B.D. Johnson. (2002). Wavelets: generalized quasi-affine and oversampled-affine frames. Thesis, Washington University in St. Louis.

39. D. Donoho and A.G. Flesia. (2001). Can recent innovations in harmonic analysis 'explain' key findings in natural image statistics? *Network Computations in Neural Systems,* Vol. 12, 371–393.

© 2008 by Taylor & Francis Group, LLC

40. D. Donoho and E. Candes. (2005). Continuous curvelet transform I: resolution of wavefront set. *Applied and Computational Harmonic Analysis,* Vol. 19, 162–197.

41. D. Donoho and E. Candes. (2005). Continuous curvelet transform II: discretization and frames. *Appl. Comput. Harmon. Anal.* Vol. 19, 198–222.

42. Curvelet website: http://www.curvelet.org/papers.html

43. A. Khodakovsky, P. Schröder, and W. Sweldens. (2000). Progressive geometry compression. *SIGGRAPH 2000,* 271–278.

44. I. Guskov, W. Sweldens, and P. Schröder. (1999). Multiresolution signal processing for meshes. *SIGGRAPH 1999,* 325-334.

45. W. Sweldens' papers: netlib.bell-labs.com/cm/ms/who/wim/papers/

46. L. Ying, L. Demanet, and E. J. Candes. (2005). 3D discrete curvelet transform. *Proceedings of SPIE—Volume 5914, Wavelets XI,* M. Papadakis, A.F. Laine, M.A. Unser (Eds.).

47. UCSD video processing page: http://videoprocessing.ucsd.edu/

6.7 Examples and exercises

Section 6.1

Example 6.1.A In this example we find the 2-level Walsh transform of the signal $\mathbf{f} = (-2, -4, 2, 6, 8, 4, 4, 2)$. First, a 1-level Haar transform is computed:

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (-3\sqrt{2}, 4\sqrt{2}, 6\sqrt{2}, 3\sqrt{2} \,|\, \sqrt{2}, -2\sqrt{2}, 2\sqrt{2}, \sqrt{2}).$$

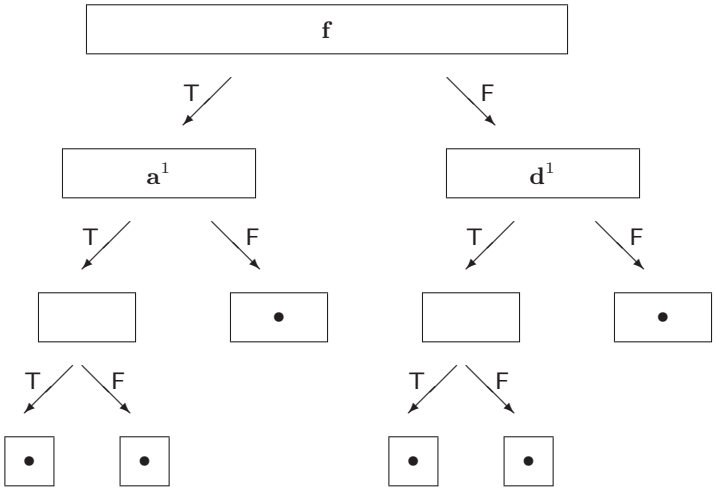
Then we compute 1-level Haar transforms of both $\mathbf{a}^1$ and $\mathbf{d}^1$, obtaining

$$(1, 9 \,|\, -7, 3 \,|\, -1, 3 \,|\, 3, 1)$$

which is the 2-level Walsh transform of $\mathbf{f}$.

Example 6.1.B [Other wavelet packet transforms] Wavelet packet transforms in general are defined as signals that result at the end nodes of a tree-diagram describing applications of a trend calculation T and a fluctuation calculation F. For example, consider the following tree diagram (where the symbol • indicates an end node signal):

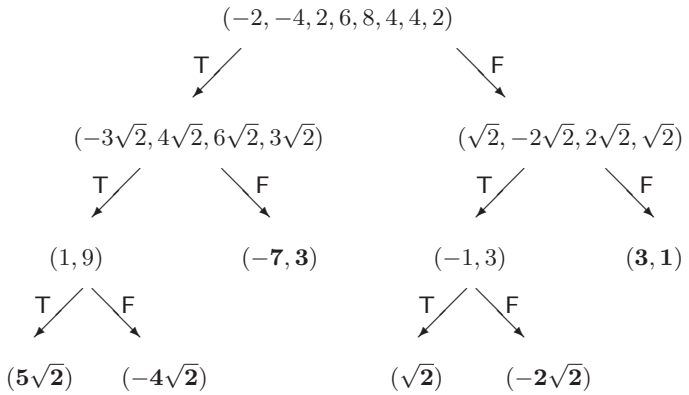
© 2008 by Taylor & Francis Group, LLC



which can also be expressed more succinctly as

$$(\mathsf{TTT}(\mathbf{f})\,|\,\mathsf{FTT}(\mathbf{f})\,|\,\mathsf{FT}(\mathbf{f})\,|\,\mathsf{TTF}(\mathbf{f})\,|\,\mathsf{FTF}(\mathbf{f})\,|\,\mathsf{FF}(\mathbf{f}))\,.$$

For instance, if $\mathbf{f} = (-2, -4, 2, 6, 8, 4, 4, 2)$ then this wavelet packet transform is computed as follows (with the end-node signals given in bold-face):



and we can write this wavelet packet transformed signal as:

$$(5\sqrt{2}\,|\,-4\sqrt{2}\,|\,-7,3\,|\,\sqrt{2}\,|\,-2\sqrt{2}\,|\,3,1).$$

Example 6.1.C In this example we show how to use FAWAV to compute a wavelet packet series of the kind described in the text, where each subsignal is transformed at every level, for a Coif30 wavelet. Suppose that our signal is obtained from plotting the formula $40\sin(12\pi x^2)$ over the interval $[0, 4]$ using 4096 points. To compute a 6-level Coif30 wavelet packet series, transforming each subsignal at

© 2008 by Taylor & Francis Group, LLC

every level, we select *Series* and *Wavelet Packet* and then specify 6 levels and a Coif30 wavelet. If we also select *Energy fraction* and specify 0.9999, then FAWAV uses 250 transform values (which it reports as "250 coefficients") to obtain a wavelet packet approximation of the signal that captures 99.99% of the signal's energy. To see the advantage for this signal of computing a wavelet packet series, if we compute a 6-level Coif30 wavelet series with energy fraction 0.9999, then FAWAV uses 397 transform values, a far greater number.

6.1.1$_s$ Given $\mathbf{f} = (2, 4, 8, 6, 2, 4, 6, 8)$, find its 2-level Walsh transform.

6.1.2 Given $\mathbf{f} = (10, 8, 4, -2, 4, 8, 10, 18)$, find its 3-level Walsh transform.

6.1.3^c_s Given the signal obtained by plotting $\sin(24\pi x^2) - \sin(12\pi x^2)$ over $[0, 2]$ using 4096 points, how many transform values (coefficients) are needed in a 6-level Coif30 wavelet packet series to capture 99.99% of the signal's energy? How many for a Coif30 wavelet series?

6.1.4^c Given the signal obtained by plotting $\sin[24\pi(2 - x)^2] + 4\sin(12\pi x^2)$ over $[0, 2]$ using 4096 points, how many transform values (coefficients) are needed in a 6-level Coif30 wavelet packet series to capture 99.99% of the signal's energy? How many for a Coif30 wavelet series?

6.1.5 Given the signal $\mathbf{f} = (2, 4, 6, 8, 16, 20, 22, 22)$, compute its wavelet packet transform defined by

$$(\mathsf{TT}(\mathbf{f}), \mathsf{TFT}(\mathbf{f}), \mathsf{FFT}(\mathbf{f}), \mathsf{TF}(\mathbf{f}), \mathsf{TFF}(\mathbf{f}), \mathsf{FFF}(\mathbf{f}))$$

and draw the tree diagram corresponding to this transform.

6.1.6 Given the signal $\mathbf{f} = (2, 0, 4, 4, 4, 2, 4, 6, 8, 12, 12, 10, 8, 8, 8, 6)$, compute its wavelet packet transform defined by the following transforms applied to $\mathbf{f}$

$$\mathsf{TTT}, \mathsf{TFTT}, \mathsf{FFTT}, \mathsf{FT}, \mathsf{TTF}, \mathsf{TFTF}, \mathsf{FFTF}, \mathsf{TFF}, \mathsf{FFF}$$

and draw the tree diagram corresponding to this wavelet packet transform.

Section 6.2

Example 6.2.A [Table 6.1] The values in the first row were obtained as follows. We began by loading the sound file `greasy.wav` and then selected *Series/Wavelet*. After specifying 4 levels, and a Coif30 wavelet, and choosing the *Threshold* option, we then clicked on the *Edit Settings* button. In the edit window that opens we selected the *Multiple Thresholds* option and entered `1/2^7:1/2^5` for the thresholds. After clicking *Apply* we then plotted the wavelet series. The entries *Sig. values* and *Bpp* were then read off from the report of the number of coefficients and the bits per point, respectively. The RMS error was obtained from selecting *Analysis/Norm difference* and using the default choices (Normalized, absolute Power 2 norm). Similar work was done to get the second row's values, except that we used the choice *Series/Wavelet packet. Note:* your results may differ slightly from ours due to differences in floating point arithmetic for various CPUs.

Example 6.2.B [Table 6.2] To obtain the data in the first row of the table, we loaded the image `Barb.bmp` and performed a 4-level Daub 9/7 wavelet transform.

© 2008 by Taylor & Francis Group, LLC

We then right-clicked on the transform image and selected *Save graph/Graph.* We then saved our transform data to a file `Barb_4_tr.fb2` in the *Data* subdirectory of the FAWAV directory. After that, we selected *Image compression workshop* from the *Graph* menu, and proceeded as follows. (*Note:* we go through this procedure of using *Image compression workshop* for consistency, since it is the only way that FAWAV has for performing wavelet packet compression.)

Within the *Image compression workshop* window we specified 4 for the number of levels, and 0.5 for the target bpp rate, and then clicked the button labelled *Encode Transform* and selected the file `Barb_4_tr.fb2` for encoding. The *Sig. values* entry was then read off from the report generated by the encoding process when it completed its encoding at the rate of almost exactly the target rate of 0.5 bpp. We then clicked on the button *Decode Transform* and selected the file `Barb_4_tr.wic` from the *Compress/Data* subdirectory of the FAWAV directory. (*Note:* when selecting this file, you might first right-click on it and check its *Properties* to verify that it has a file size of 16 kb, which is a 16:1 compression of the 256 kb file `Barb.bmp`.) The decoded transform was saved to the file `Barb_4_tr_quant.fb2` in the *Compress/Data* subdirectory. Finally, we returned to the 2D-form containing the images and right-clicked on one of the images, followed by selecting *Load/Graph* and loaded the file `Barb_4_tr_quant.fb2`. After inverse transforming this image, and 8-bit quantizing the resulting image [by deleting one of the transform images, and then using *Graph/Quantize (8-bit)*], we had our decompressed image. We obtained the PSNR by selecting *Analysis/Norm difference* and choosing the *PSNR* option. The second row of the table was obtained by repeating this work, but using a wavelet packet transform and its inverse. The third and fourth rows were obtained in the same way as the first and second rows, except that 0.25 bpp was used as the target rate for compression.

Example 6.2.C [**Figure 6.1**] The images in the Figure were obtained during the process of creating the decompressions of the 16:1 compressions of `Barb.bmp` described in the previous example, and zooming in twice on a region around the fold in Barb's scarf (using the same center pixel values for each zooming). The PSNRs computed by FAWAV are obtained just for the region selected in the zooms (using the first specified graph to determine the zoomed region, so be sure that you have zoomed on exactly the same pixel coordinates for each image).

Example 6.2.D [**Figure 6.2**] The image in (a) was created by performing a 2-level Daub 9/7 transform of the image `Barb.bmp` and then plotting

```
g2 a + 950(abs(g2)>8)(1-a)
+ 475(abs(g2)<8)(1-a)
\a = (x<c)(y<c)\c = -.5 + 1/2^2
```

The value 950 is the maximum magnitude for the transformed image (obtained using *Analysis/Statistics*), so this formula displays the brightest white for transform values that exceed 8 in magnitude, and a dull gray for those which do not, and it leaves the trend image values unchanged from their original values. The image in (b) was produced in the same way, except a wavelet packet transform was used.

6.2.1$_{\mathrm{s}}^{\mathrm{c}}$ Compute 5-level Coif30 compressions of `alfalfa_22050.wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to

© 2008 by Taylor & Francis Group, LLC

Table 6.1. Can you hear any differences between the compressed signals and the original?

6.2.2^{c} Compute 5-level Coif30 compressions of `denoised_thrush.wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to Table 6.1.

6.2.3^{c} Compute 5-level Coif30 compressions of `Call(Jim).wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to Table 6.1.

6.2.4$^{c}_{s}$ The file `noisy_osprey_clip.wav` is a noisy version of `osprey_clip.wav`. Compute RMS errors for threshold denoisings with both wavelet and wavelet packet transforms. How much reduction of RMS error is obtained through this denoising? Which denoising sounds better, and why?

6.2.5^{c} Load `Call(Jim).wav` and simulate a noisy version by plotting

```
g1(x) + rang(0)
```

Perform threshold denoising using a wavelet transform and a wavelet packet denoising. Which denoising has smaller RMS? Why? Which sounds better?

6.2.6$^{c}_{s}$ Produce a table like Table 6.2, but use the `Boat.bmp` image.

6.2.7 Produce a table like Table 6.2, but use the `Airfield.bmp` image.

6.2.8 Produce a table like Table 6.2, but use the `Peppers.bmp` image.

6.2.9 Compare J2K, ASWDR, and WSQ for compressing `fingerprint_1.bmp` at a 10:1 compression ratio. By compare, we mean in terms of PSNR and subjective visual inspection of the whole images and zooms about the coordinates $(256, 256)$.[7] Which method produces the best results?

6.2.10^{c} Repeat the previous exercise, using 20:1 compression.

6.2.11^{c} Repeat Exercise 6.2.9, using the `Fingerprint_2.bmp` image and zoom coordinates $(128, 128)$.

6.2.12^{c} Repeat Exercise 6.2.10, using the `Fingerprint_2.bmp` image and zoom coordinates $(128, 128)$.

Section 6.3

Example 6.3.A [**Figure 6.3**] The graph in (a) was obtained by plotting

```
2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w
\w=1/16\u=ax\a=2^(m/6)
\m=0
```

[7] The J2K compression can be done with ImageAnalyzer, and its decompression saved as a `bmp` file for computing PSNR with FAWAV. Similarly, the WSQ compression can be done with WSQ Viewer.

© 2008 by Taylor & Francis Group, LLC

over the interval $[-0.5, 0.5]$ using 4096 points. The top graph in (b) was then obtained by selecting *Transform/Fourier,* clicking the *Plot* button, and retaining only the first graph plotted (the real part of the DFT, since the imaginary part is essentially zero) and changing the X-interval to $[-75, 75]$. The rest of the graphs in (b) were obtained by successively plotting the formula above (changing `\m=0` to `\m=2`, `\m=4`,..., `\m=8`), and then plotting Fourier transforms.

Example 6.3.B [Figure 6.4] The image in (a) was obtained by plotting

```
sin(40pi x)e^{-100pi (x-.2)^2}+
[sin(40pi x)+2cos(160pi x)]e^{-50pi (x-.5)^2}
+ 2sin(160pi x)e^{-100pi(x-.8)^2}
```

over the interval $[0, 1]$ using 2048 points, selecting *Scalogram/Analysis,* and computing a Mexican hat scalogram with *Width* specified as 1/16. The image in (b) was obtained by plotting

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k-1)>.5))
+.1sin(12pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x-2
```

over the interval $[-5, 5]$ using 1024 points, changing the plotting style to *Lines,* and then computing a Mexican hat scalogram with *Width* specified as 2.

6.3.1$^{c}_{s}$ Compute Blackman windowed spectrograms for the following three shifted and scaled Mexican hat wavelets:

```
2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=a(x+.25)\a=2^(m/6) \m=8

2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=ax\a=2^(m/6) \m=32

2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=a(x-.25)\a=2^(m/6) \m=16
```

over the interval $[-0.5, 0.5]$ using 1024 points [and displaying the spectrograms with *Display style* setting *Linear (global)*]. How do these spectrograms relate to the time and frequency decomposition given by a Mexican hat CWT?

6.3.2^{c} For the following test signal

```
[sin(80pi x)-cos(40pi x)]e^{-100pi (x-.2)^2}+
[sin(160pi x) + cos(80pi x)]e^{-50pi (x-.5)^2}
+ sin(80pi x)e^{-100pi(x-.8)^2}
```

graphed over $[0, 1]$ using 4096 points, plot its Mexican hat scalogram using 6 octaves, 42 voices, and a width of 0.05. Explain the relationship between the features of the scalogram and the frequencies of the sines and cosines in the signal's formula.

6.3.3^{c} For the following simulated ECG signal

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k+1)>.5))
+.1sin(16pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x+2
```

© 2008 by Taylor & Francis Group, LLC

graphed over the interval $[-5, 5]$ using 8192 points, plot its Mexican hat scalogram using 8 octaves, 16 voices, and width 2.

6.3.4^{c} For the following simulated ECG signal

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1))
+.2sin(16pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x+2
```

graphed over the interval $[-5, 5]$ using 8192 points, plot its Mexican hat scalogram using 8 octaves, 16 voices, and width 2.

Section 6.4

Example 6.4.A [Figure 6.5] The image in (a) was produced by graphing the same function as in Example 6.3.B, then selecting *Analysis/Scalogram*, and choosing a *Gabor (complex)* scalogram. The scalogram was plotted using 8 octaves and 16 voices, and a width parameter of 1 and freq. parameter of 5. The image in (b) was obtained by computing a Blackman windowed spectrogram of the signal.

Example 6.4.B [Figure 6.6] The image in (a) was obtained by loading the sound file `Call(Jim).wav` and computing a Blackman windowed spectrogram. The image in (b) was computed using a *Gabor (complex)* scalogram using 4 octaves and 16 voices, and a width parameter of 1/8 and freq. parameter of 10.

Example 6.4.C [Figure 6.7] The graph in (a) was obtained by loading the sound file `Call(Jim).wav`, then changing the X-interval to $.09, .09 + .371519274376417/2$, and then clipping out the displayed graph (right-clicking and selecting *Clip* from the popup menu). The spectrum in (b) was obtained from (a) by selecting *Transforms/Fourier,* and choosing the options *Amp/Phase* and interval type $[0, L] \rightarrow [-A, A]$. We then removed the phase graph (graph 2) from the resulting transform, and changed the X-interval to $[0, 992]$ and Y-interval to $[-1, 3]$.

Example 6.4.D [Figure 6.8] The image in (a) was obtained via a Blackman windowed spectrogram of the sound file `Buenos_aires_Madonna_lyrics.wav`. The image in (b) was obtained by computing a scalogram of type *Gabor (complex)* using 3 octaves and 85 voices, and a width parameter of 0.1 and freq. parameter of 20, and changing the display style to *Log (global)*.

6.4.1^{c} Compute spectrograms and scalograms for the word "call" from each of the following 10 recordings:

`call back 1.wav, call back 2.wav,..., call back 10.wav.`

Can you formulate any conjectures about formants of various speakers (e.g., male and female, or Native English and Foreign, etc.)?

6.4.2 Do a time-frequency analysis of the sound clip `chaiya chaiya clip.wav`. Does the Multiresolution Principle apply here? *Note:* The spectrogram is best viewed with AUDACITY. The *Gabor (complex)* scalograms from FAWAV will need to be constructed from clips of the audio file using the following settings: 3 Octaves, 32 voices, width 0.05, and freq. 10.

© 2008 by Taylor & Francis Group, LLC

Section 6.5

Example 6.5.A [**Figure 6.9**] The spectrogram in (a) was generated by loading the sound file `el_matador_percussion_clip.wav` and then selecting *Analysis/Spectrogram* and performing a Blackman windowed spectrogram with the default settings. To produce (b), the sound file `Buenos Aires percussion clip.wav` was processed in a similar way.

Example 6.5.B [**Figure 6.10**] The processed spectrogram at the top of the figure was produced from the spectrogram shown in Figure 6.9(a) by selecting *Graph/Plot* and then plotting the formula

```
g1(2500 < y < 4500)
```

The pulse train shown at the bottom was then generated by selecting *Graph* and choosing *Percussion scalogram.*

Example 6.5.C [**Figure 6.11**] The percussion scalogram was generated from the pulse train described in the previous example. You select a *Gabor (complex)* type of scalogram and then enter the following data:

```
Octaves:  4          Voices:  64

Width:  0.5          Freq.:  0.5
```

and plot the scalogram.

Example 6.5.D [**Figure 6.12**] The percussion scalogram in this figure was generated in the following way. First, compute the Blackman-windowed spectrogram of the sound file `Buenos Aires percussion clip.wav`. Second, from the spectrogram menu select *Graph/Plot* and plot the following function:

```
g1(2000 < y < 3000)
```

Third, select *Graph/Percussion scalogram* from the spectrogram menu. In the *scalogram* window that opens up, you specify a *Gabor (complex)* type of scalogram, enter the following data

```
Octaves:  5          Voices:  51

Width:  2            Freq.:  1
```

and plot the scalogram.

6.5.1$^{c}_{s}$ Use the percussion scalogram method to analyze the rhythm in the audio file `Mingus_Dizzy_(Clip_1).wav`.

6.5.2$^{c}_{s}$ Use the percussion scalogram method to analyze the rhythm in the audio file `Mingus_Dizzy_(Clip_2).wav`.

6.5.3$^{c}_{s}$ Use the percussion scalogram method to analyze the rhythm in the audio file `Slapshift_(Clip_1).wav`.

© 2008 by Taylor & Francis Group, LLC

6.5.4$^{c}_{s}$ Use the percussion scalogram method to analyze the rhythm in the audio file `Slapshift_(Clip_2).wav`. [Hint: You may want to only analyze frequency data above 1500 Hz.]

6.5.5^{c} Use the percussion scalogram method to analyze the rhythm in the audio file `Conga_solo_clip.wav`.

6.5.6^{c} Use the percussion scalogram method to analyze the rhythm in the audio file `brazil_clip.wav`.

6.5.7^{c} Use the percussion scalogram method to analyze the rhythm in the audio file `brazil_medley_clip.wav`.

© 2008 by Taylor & Francis Group, LLC

Appendix A

Projects

The work of applying wavelets and time-frequency analysis to practical problems has only just begun. There are lots of problems to work on. I have compiled a list of some of the topics that the reader might find interesting to pursue. Naturally, these ideas originate from my own research.

Some of these ideas may already have been worked on by me and my student collaborators as part of the NSF REU (Research Experience for Undergraduates) held at UWEC, known as SUREPAM (www.uwec.edu/surepam/). If you are interested in these ideas, please contact me at walkerjs@uwec.edu so that I can tell you if any work has been done already. Even if it has, maybe we can collaborate. If you do work that you intend to publish in some form, I would very much like to hear from you, and would be most happy to assist you in any way that I can. As work develops with these and other future projects, I hope to create an archive of final results—PDF files or other formats at the primer website—so let's keep in touch!

The projects are in three areas: (1) Music, (2) Noise Removal from Audio, and (3) Image Processing.

A.1 Music

The application of mathematics to musical theory is a hot topic in both the math and music worlds. I have published two papers in this area with students [1,2]. The most important background on the subject can be found in an article that I have written with a music professor [3], and an article on musical rhythm that I have written with two student collaborators [4]. Based on these

© 2008 by Taylor & Francis Group, LLC

articles, here are four problems that could be worked on.

1. Detailed analysis of a passage of the song *Buenos Aires* from the musical *Evita* using the methods discussed in [3,4].

2. Detailed analysis of other musical passages. Some possibilities include: a passage from Stravinsky's *Rite of Spring,* or a passage from Copland's *Appalachian Spring,* or a passage from Messiaen's *Catalogue d'Oiseaux* (including comparison with bird song), or other passages from jazz, or world folk music, or other musical styles.

3. Use the inverse Gabor transform to synthesize new musical passages.

4. Develop and implement (either in MAPLE®, or MATLAB®, or VISUAL BASIC®) the new method of Reassigned Short-time Fourier Transforms [5] and apply it to analyzing music using the methods described in [3,4].

A.2 Noise removal from audio

The application of the method of Gabor transforms to audio noise removal is described in [6]. Based on that article, here are three problems that could be worked on.

1. Generalizing the method described in [6] to noise that has different standard deviations within different frequency ranges (one such noise, so-called "pink noise," occurs in real, environmental, background noise).

2. Generalizing the method described in [6] to noise with changing standard deviation over time (non-stationary noise). This is a *very* difficult problem, but even partial results would be of great interest.

3. Investigating "soft thresholding" (see the discussion of noise removal in Section 5.10 and in [3]) for generalizing the noise removal method in [6].

A.3 Wavelet image processing

We introduced some ideas for image processing in Chapter 4. Here are three projects that extend some of those ideas.

1. Develop an improved algorithm, and software, for both automatic and user-guided clutter removal from STM images (see Subsection 4.8.4).

2. Develop a model for the clutter and prove that the algorithm from item 8 is an effective clutter removal method.

3. This project deals with *super-resolution image enhancement.* Develop an image enhancement algorithm based on the Principles A and B of ASWDR (see Section 4.5). Use those principles to project up to finer

© 2008 by Taylor & Francis Group, LLC

scale wavelet values, followed by inverse wavelet transform, to produce a higher resolution image.

A.4 References

1. J.F. Alm and J.S. Walker. (2002). Time-frequency analysis of musical instruments. *SIAM Review,* Vol. 44, 457–476.
2. J.S. Walker and A.J. Potts. (2003). Time-frequency spectra of music. *Advances in Analysis: Proceedings of 4th International ISAAC Congress.* World Scientific, Singapore, 487–493.
3. J.S. Walker and G.W. Don. (2006). Music: a time-frequency approach. Preprint. Available at http://www.uwec.edu/walkerjs/Primer/Papers
4. X. Cheng, J.V. Hart, and J.S. Walker. (2007). Time-frequency analysis of musical rhythm. Preprint. Available at

   http://www.uwec.edu/walkerjs/media/TFAMR.pdf

5. T. Gardner and M. Magnasco. (2006). Sparse time-frequency representations. *Proc. Nat. Acad. Sci.,* Vol. 103, 6094–6099. Available at

   http://www.pnas.org/cgi/reprint/103/16/6094.pdf

6. J.S. Walker and Y.-J. Chen. (2006). Denoising Gabor transforms. Preprint. Available at

   http://www.uwec.edu/walkerjs/media/DGT.pdf

© 2008 by Taylor & Francis Group, LLC

Appendix B

Selected exercise solutions

B.1 Introduction

Just a brief word of introduction. We have endeavored to provide accurate solutions to all of the selected problems for Chapters 2 to 6. If you find any errors, please contact me via email at walkerjs@uwec.edu and I will post corrections on an *Errata* link on the book's website.

B.2 Chapter 2

2.1.1 (a) $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (3\sqrt{2}, 6\sqrt{2}, 3\sqrt{2} \,|\, -\sqrt{2}, 0, \sqrt{2})$
(c) $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (1.5\sqrt{2}, 3\sqrt{2}, 2\sqrt{2}, \sqrt{2} \,|\, -0.5\sqrt{2}, 0, 0, 0)$

2.1.2 (a) $\mathbf{f} = (2, 2, 0, -2, 3, 3, 0, -2)$ (c) $\mathbf{f} = (5, 1, 1, 3, 3, 1, 0, 0)$

2.1.3 (a) $\widetilde{\mathbf{f}} = (2, 2, 3, 3, 4.5, 4.5, 6, 6)$, largest error $= 0.5$.
(c) $\widetilde{\mathbf{f}} = (0, 0, -2, -2, -1, -1, 2, 2)$, largest error $= 2$.

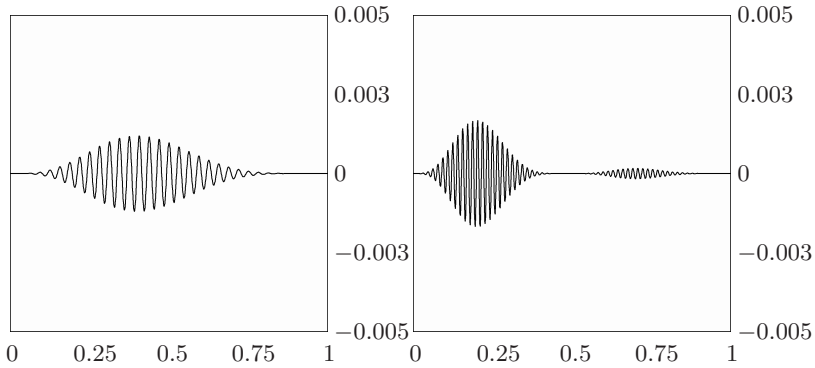
2.1.5 (b) The original signal and its 1-level Haar transform are shown in Figure B.1.

2.2.1 (a) $\mathcal{E}_{\mathbf{a}^1} = 108$, $\mathcal{E}_{\mathbf{d}^1} = 4$, $\mathcal{E}_{\mathbf{f}} = 112 = 108 + 4$. (c) $\mathcal{E}_{\mathbf{a}^1} = 32.5$, $\mathcal{E}_{\mathbf{d}^1} = 0.5$, $\mathcal{E}_{\mathbf{f}} = 33 = 32.5 + 0.5$.

2.2.3 (a) 16.7% (c) 37.5%

2.2.5 (b) $\mathcal{E}_{\mathbf{a}^1} = 1.91685\cdots \times 10^{-4}$, $\mathcal{E}_{\mathbf{d}^1} = 1.86057\cdots \times 10^{-6}$, $\mathcal{E}_{\mathbf{f}} = 1.935457 \times$

© 2008 by Taylor & Francis Group, LLC



(a) Signal (b) 1-level Haar transform

FIGURE B.1
Solution to 2.1.5(b).

$10^{-4} = \mathcal{E}_{\mathbf{a}^1} + \mathcal{E}_{\mathbf{d}^1}$ to an accuracy slightly better than 6×10^{-19}.

2.2.7 (b) The 1-level, 2-level, and 3-level transforms are respectively:

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (-16\sqrt{2}, 8\sqrt{2}, 48\sqrt{2}, 96\sqrt{2} \,|\, 0, -24\sqrt{2}, 0, 0)$$
$$(\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (-8, 144 \,|\, -24, -48 \,|\, 0, -24\sqrt{2}, 0, 0)$$
$$(\mathbf{a}^3 \,|\, \mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (68\sqrt{2} \,|\, -76\sqrt{2} \,|\, -24, -48 \,|\, 0, -24\sqrt{2}, 0, 0).$$

2.3.1 (a) -2 (c) 8

2.3.5 $(0.5\sqrt{2}, 0.5\sqrt{2}, 0, 0, \ldots, 0)$ and $(0.5\sqrt{2}, -0.5\sqrt{2}, 0, 0, \ldots, 0)$

2.4.1 (a) $\mathbf{f} + \mathbf{g} = (3, 5, 1, 7)$, $\mathbf{f} - \mathbf{g} = (1, 1, 3, 1)$, $3\mathbf{f} - 2\mathbf{g} = (4, 5, 8, 6)$
(c) $\mathbf{f} + \mathbf{g} = (0, 1, 1, 2, 0, 2)$, $\mathbf{f} - \mathbf{g} = (-2, -3, 1, 0, 2, 0)$, $3\mathbf{f} - 2\mathbf{g} = (-5, -7, 3, 1, 5, 1)$

2.4.2 (a) $\mathbf{A}^1 = (3, 3, 6, 6, 3, 3)$, $\mathbf{D}^1 = (-1, 1, 0, 0, 1, -1)$
(c) $\mathbf{A}^1 = (1.5, 1.5, 3, 3, 2, 2, 1, 1)$, $\mathbf{D}^1 = (-0.5, 0.5, 0, 0, 0, 0, 0, 0)$

2.4.4 (a) $\mathbf{A}^1 = (1.5, 1.5, 1, 1, 3, 3, 3.5, 3.5)$, $\mathbf{D}^1 = (0.5, -0.5, 2, -2, -1, 1, -0.5, 0.5)$
(c) $\mathbf{A}^1 = (1.5, 1.5, -1, -1, 3.5, 3.5, 2, 2)$, $\mathbf{D}^1 = (-0.5, 0.5, 0, 0, 0.5, -0.5, 0, 0)$

2.4.6 (a) $\mathbf{A}^2 = (1.25, 1.25, 1.25, 1.25, 3.25, 3.25, 3.25, 3.25)$,
$\mathbf{D}^2 = (.25, .25, -.25, -.25, -.25, -.25, .25, .25)$
(c) $\mathbf{A}^2 = (.25, .25, .25, .25, 2.75, 2.75, 2.75, 2.75)$,
$\mathbf{D}^2 = (1.25, 1.25, -1.25, -1.25, 0.75, 0.75, -0.75, -0.75)$

2.5.1 The graphs are shown in Figure B.2; 52 transform values were used to produce the signal in Figure B.2(d), hence a 19.7:1 compression ratio. Any threshold less than 0.5 will produce at least 99.99% of energy; using 0.49 we obtained a signal that has a maximum error of 0.125. [*Note:* since the signal is integer-valued, this

© 2008 by Taylor & Francis Group, LLC

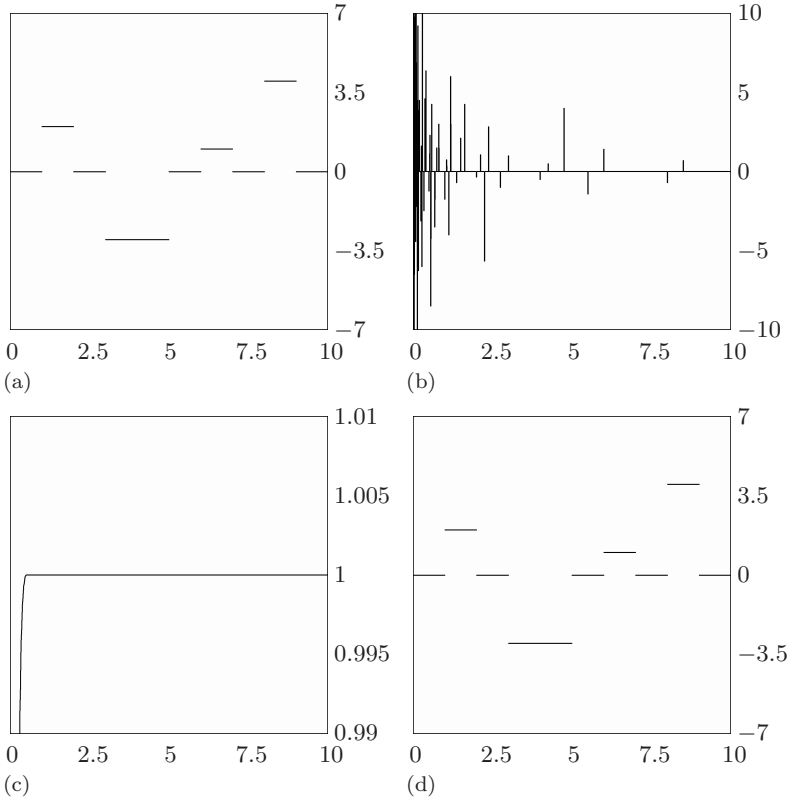


FIGURE B.2
(a) Signal. (b) 10-level Haar transform. (c) energy map of Haar transform. (d) 19.7:1 compression, 100% of energy.

error could be completely eliminated by rounding to nearest integers.]

2.5.2 The graphs are shown in Figure B.3. To get 100% energy requires *all* of the coefficients (no compression) so we did not graph it. Instead, we found that a threshold of 1.245 will retain 65 transform values (a compression ratio of 15.8:1) and produces the graph shown in Figure B.3(d). The maximum error between this signal and the original signal is 0.604.

2.5.3 For (a) we get a sup-norm difference of 0.74, while for (b) we get 2.66×10^{-15}. Clearly the series for (b) performs the best. The reason is that the function in (b) is a step function so all of the Haar transform values are 0, except for a small number corresponding to Haar wavelets whose supports overlap the jumps in the step function. Those relatively few high-magnitude coefficients are all included by specifying the highest 50, hence the extremely small error (due to the roundoff error that always occurs with digital calculations). The function in (a) is not a step function so it has many more non-zero transform values.

2.6.2 It looks like a sequence of random numbers (just as the signal from problem

© 2008 by Taylor & Francis Group, LLC

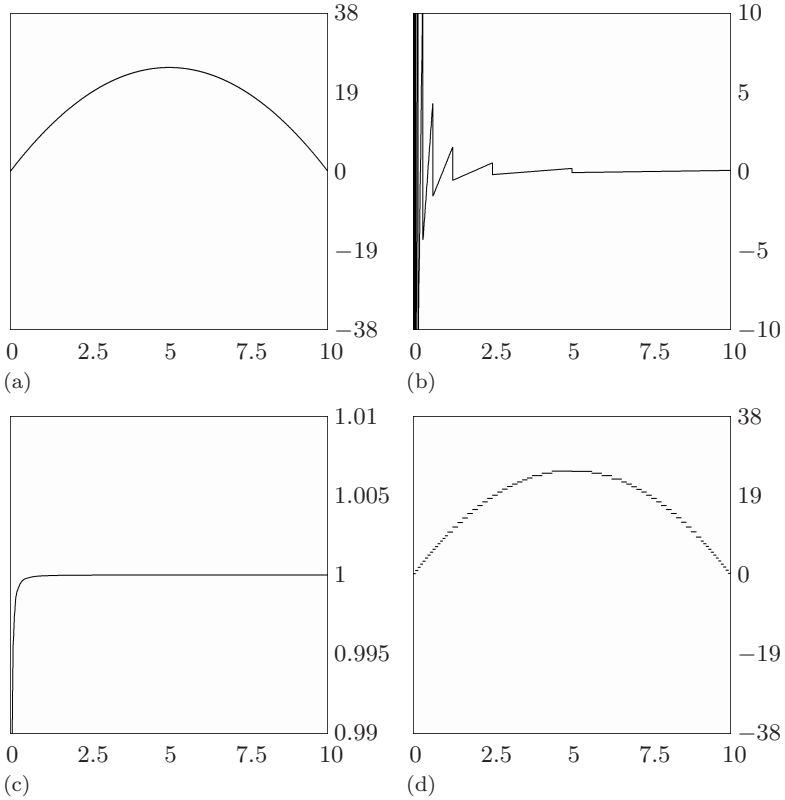


FIGURE B.3
(a) Signal. (b) 10-level Haar transform. (c) energy map of Haar transform. (d) 15.8:1 compression of Signal, 99.99% of energy.

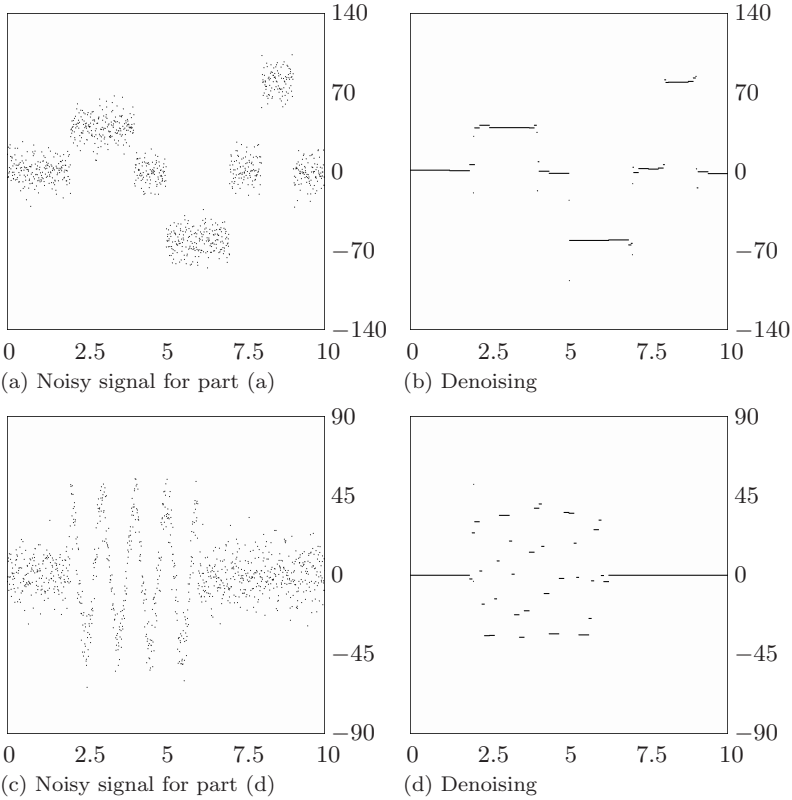
2.6.1 looks random) and, again just like the signal from 2.6.1, it sounds like the static one hears in radio transmissions.

2.6.3 Noisy signals are shown for parts (a) and (d) in Figure B.4. To denoise these signals a threshold of 35 was used in both cases. The denoised signals are also shown in Figure B.4. The denoising for the step function in (a) was the best (most representative of the original signal). The main reason is that the underlying signal for (a) is a step function which is best for Haar transforms.

B.3 Chapter 3

3.1.1 $\mathbf{V}_1^2 = \alpha_1\mathbf{V}_1^1 + \alpha_2\mathbf{V}_2^1 + \alpha_3\mathbf{V}_3^1 + \alpha_4\mathbf{V}_4^1$ where each $\mathbf{V}_k^1$, $k = 2, 3, 4$, is a translate of $\mathbf{V}_1^1$ by $2(k-1)$ time-units. It follows that $\mathbf{V}_4^1$ is a translate by 6 units with a support of 4 units, and therefore $\mathbf{V}_1^2$ has a support of 10 time-units. $\mathbf{V}_2^2 = \alpha_1\mathbf{V}_3^1 + \alpha_2\mathbf{V}_4^1 + \alpha_3\mathbf{V}_5^1 + \alpha_4\mathbf{V}_6^1$ and therefore it has a support of 10 time-units. Because its support begins with the support of $\mathbf{V}_3^1$, which is a translation of $\mathbf{V}_1^1$ by

© 2008 by Taylor & Francis Group, LLC



(a) Noisy signal for part (a)
(b) Denoising
(c) Noisy signal for part (d)
(d) Denoising

FIGURE B.4
Denoising for Exercise 2.6.3.

4 units, it follows that V_2^2 is a translation of V_1^2 by 4 units. Similar calculations show that, in general, $\mathbf{V}_m^2$ has a support of 10 time-units and is a translation of $\mathbf{V}_1^2$ by $4(m-1)$ time-units.

3.1.3 No, it equals 0.896575 . . .

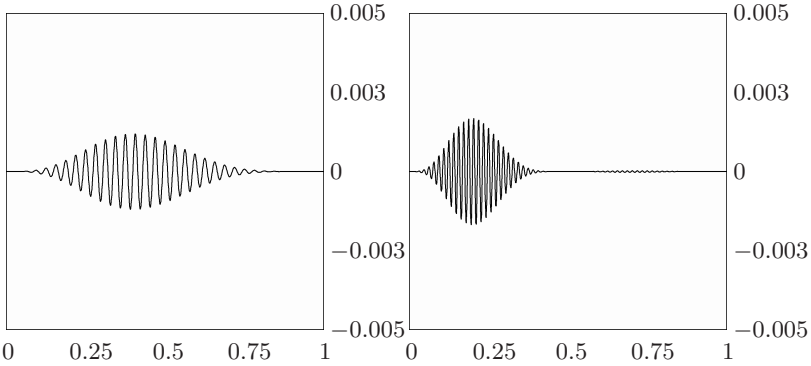
3.1.5 See Figure B.5.

3.1.7 Maximum errors: $\mathbf{A}^1$: 2.259×10^{-4}, $\mathbf{A}^2$: 1.389×10^{-3}, $\mathbf{A}^3$: 5.877×10^{-3}, $\mathbf{A}^4$: 1.572×10^{-2}.

3.1.10 The graphs are shown in Figure B.6 below. A threshold of 3.5 retains 17 transform values (a compression ratio of 60:1) and produces the graph shown in Figure B.3(d). The maximum error between this signal and the original signal is 0.9357. [Note: Using 65 transform values, as with the Haar case, we obtained a maximum error of 0.0874, which is several times smaller than the maximum error of 0.604 for the Haar case.]

3.2.2 We found that 1, 334, 877.16 equals both the energy of $\mathbf{f}$ and its 3-level

© 2008 by Taylor & Francis Group, LLC



(a) Signal (b) 1-level Daub4 transform

FIGURE B.5
Solution to 3.1.5(b).

Daub4 transform. [*Note:* energies of signals are found by selecting *Statistics* from the *Analysis* menu and computing statistics for the desired signal.]

3.2.4 We obtained the following results

	Daub4	*Haar*
(a)	75.6%	72.1%
(d)	85.4%	86.7%

3.3.1 We have

$$g(t_{2m-1+k}) = g(t_{2m-1}) + g'(t_{2m-1})(kh) + \frac{1}{2}\, g''(t_{2m-1})(k^2h^2) + \mathcal{O}(h^3).$$

Hence,

$$\begin{aligned}\mathbf{f}\cdot\mathbf{W}_m^1 &= g(t_{2m-1})\{\beta_1+\beta_2+\beta_3\cdots+\beta_6\}\\ &\quad + g'(t_{2m-1})\{0\beta_1+1\beta+2+2\beta_3+\cdots+5\beta_6\}\\ &\quad + \frac{1}{2}\, g''(t_{2m-1})\{0^2\beta_1+1^2\beta_2+2^2\beta_3+\cdots+5^2\beta_6\}+\mathcal{O}(h^3)\\ &= \mathcal{O}(h^3).\end{aligned}$$

3.3.3 The maximum errors are $\mathbf{A}^1 : 3.0078\times10^{-5}$, $\mathbf{A}^2 : 3.5018\times10^{-4}$, $\mathbf{A}^3 : 2.7638\times10^{-3}$, $\mathbf{A}^4 : 1.4618\times10^{-2}$.

3.3.9 The minimum number of terms for 99.99% of energy are

Levels:	1	2	3	4	5	6
Daub4:	335	192	162	161	161	160
Daub6:	334	167	99	90	90	92
Daub8:	333	167	85	68	70	70

© 2008 by Taylor & Francis Group, LLC

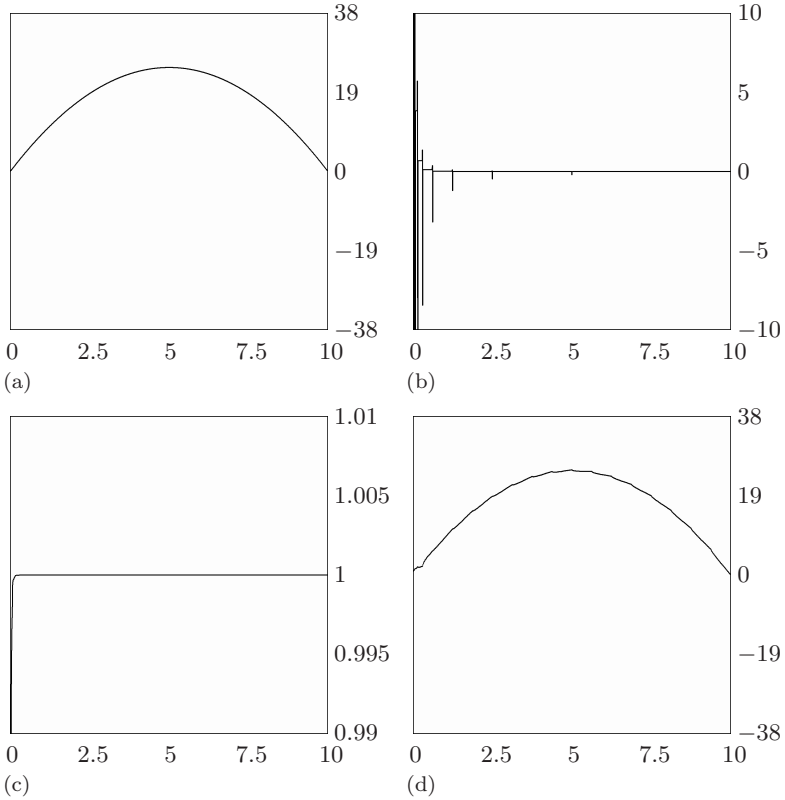


FIGURE B.6
(a) Signal. (b) 10-level Daub4 transform. (c) energy map of Daub4 transform. (d) 60:1 compression of Signal, 99.99% of energy.

3.3.12 The minimum number of terms for 99.99% of energy are

Levels:	1	2	3	4	5	6
Coif6:	334	188	160	156	155	157
Coif18:	333	168	85	52	55	54
Coif30:	333	168	85	42	44	38

3.3.16 The Coif$J$ maximum errors are: *Coif6:* 2.4296×10^{-7}, *Coif12:* 2.4842×10^{-10}, *Coif18:* 5.5743×10^{-13}, *Coif24:* 3.8137×10^{-13}, *Coif30:* 4.1636×10^{-13}. The Daub$J$ maximum errors are: *Daub4:* 3.3445×10^{-3}, *Daub6:* 4.3118×10^{-3}, *Daub8:* 5.3033×10^{-3}, *Daub10:* 6.2972×10^{-3}, *Daub12:* 7.2894×10^{-3}, *Daub14:* 8.2789×10^{-3}, *Daub16:* 9.2653×10^{-3}, *Daub18:* 1.0249×10^{-2}, *Daub20:* 1.1230×10^{-2}.

3.4.1(d) The graphs for part (d) are shown in Figure B.7. The threshold used was 2.0, which retained only 19 coefficients of the transform, hence a $1024 : 19 \approx 54 : 1$

© 2008 by Taylor & Francis Group, LLC

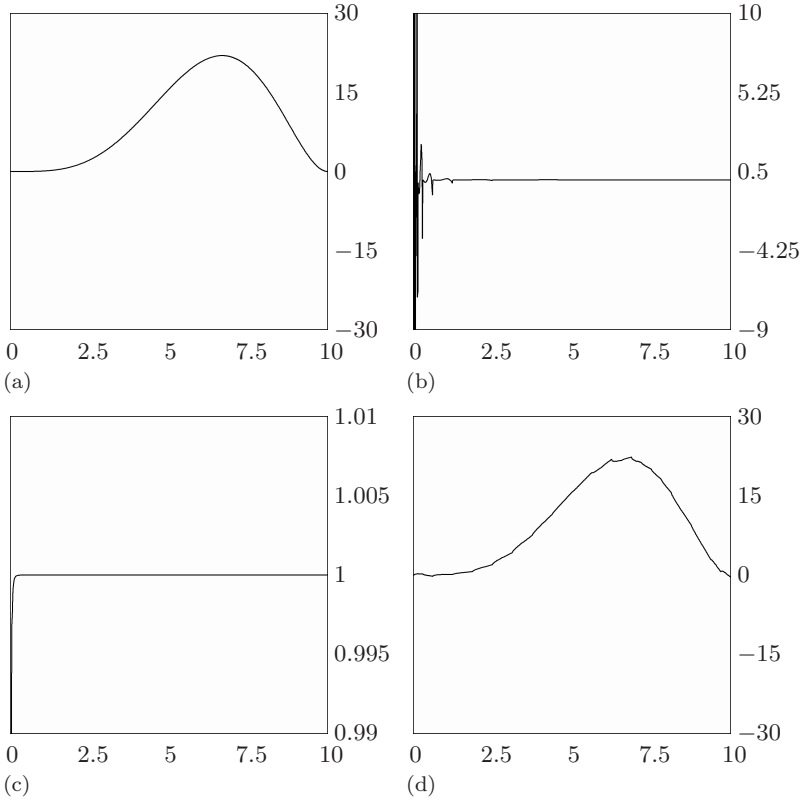


FIGURE B.7
(a) Signal. (b) 10-level Daub4 transform. (c) energy map of Daub4 transform. (d) 54:1 compression, 99.99% of energy.

compression ratio. The maximum error was 0.4411.

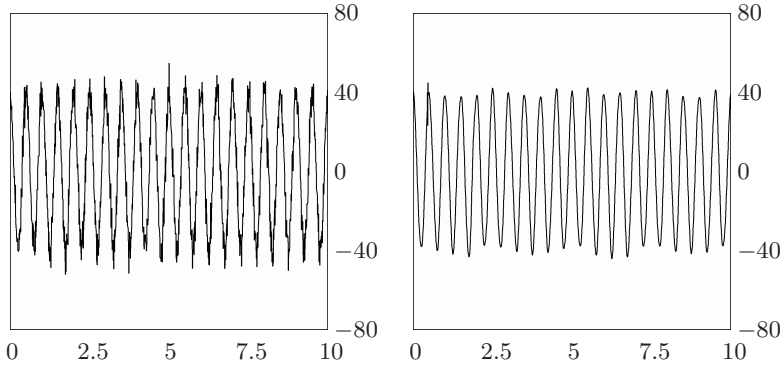
3.4.5(d) With 6 levels there was a minimum number of 17 transform values.

3.4.7(d) With 7 levels there was a minimum number of 9 transform values.

3.5.2 The entropy is $\sum_{k=1}^{N} p_k \log_2(1/p_k) = \sum_{k=1}^{N} (1/N) \log_2 N = \log_2 N$.

3.5.4 Using the rule-of-thumb of (entropy) + 0.5 we obtained these estimates for the signal `alfalfa_2.wav` (which is available from the FAWAV website): Original signal: $\approx$ 11.9 bpp. 13-level Coif30 transform (dead-zone histogram): $\approx$ 10.3 bpp. 4-level Coif30 transform (with different quantization on the trend and fluctuation): $\approx$ 7.7 bpp.

© 2008 by Taylor & Francis Group, LLC



(a) Signal (b) Denoised signal

FIGURE B.8
Solution to 3.6.1(b).

3.5.6 For problem 3.5.4, using the signal `alfalfa_2.wav` we obtained these errors:

	Sup-norm diff.	Rel. 2-Norm diff.
13-level, threshold `1/2^16`	0.622	5.66×10^{-5}
4-level, thresholds `1/2^16, 1/2^12`	11.68	1.02×10^{-3}

Note: for both compressions the sound of the signals was indistinguishable from the original recording.

3.6.1(b) The signal and denoised signal (using a threshold of 16) are shown in Figure B.8. This was the best of the denoisings, due to the lack of any large jumps in the signal values. The denoising for (d), with threshold 16, was the next best.

3.6.4 The RMS error (2-norm difference) for the noisy signal was 501.87 while for the denoised signal it was 379.60, a 24% reduction.

3.7.2 The maximum errors are $\mathbf{A}^1 : 1.32\times10^{-4}$, $\mathbf{A}^2 : 6.15\times10^{-4}$, $\mathbf{A}^3 : 2.31\times10^{-3}$, $\mathbf{A}^4 : 1.17 \times 10^{-2}$.

3.7.5 The minimum number of terms are listed in the following table.

Levels:	1	2	3	4	5	6
Terms:	335	206	206	233	246	220

3.7.6 To obtain the result we compute

```
20u^2(1-u)^4 cos(12pi u) \u = x
```

over the interval $[0, 1]$ using $16,384$ points. Then we perform a 1-level `DD 5/3 (2,2)` transform. For this transform, we change the X-interval to $[0, .5]$, then clip out the displayed trend, and then plot

```
20u^2(1-u)^4 cos(12pi u) \u = 2x
```

© 2008 by Taylor & Francis Group, LLC

(note the change from `\u = x` to `\u = 2x`). Finding the Sup-norm difference between these two graphs yields a maximum error of 5.92×10^{-7}.

3.8.2 The maximum errors are $\mathbf{A}^1 : 2.74\times10^{-6}$, $\mathbf{A}^2 : 5.36\times10^{-5}$, $\mathbf{A}^3 : 7.91\times10^{-4}$, $\mathbf{A}^4 : 9.63 \times 10^{-3}$.

3.8.5 The minimum number of terms are listed in the following table.

Levels:	1	2	3	4	5	6
Terms:	333	168	85	64	67	65

3.8.6 To obtain the result we compute

```
20u^2(1-u)^4 cos(12pi u) \u = x
```

over the interval $[0, 1]$ using $16,384$ points. Then we perform a 1-level `Daub 9/7` transform. For this transform we change the X-interval to $[0, .5]$, then clip out the displayed trend, and then plot the function

```
sqr(2)20u^2(1-u)^4 cos(12pi u) \u = 2x
```

(note the change from `\u = x` to `\u = 2x` and the factor `sqr(2)`). The Sup-norm difference between these two graphs yields a maximum error of 7.08×10^{-7}.

B.4 Chapter 4

4.1.1(a) The 1-level Haar transform array is

$$\begin{pmatrix} -4 & 1 & 0 & -1 \\ 5 & 9 & -3 & 1 \\ 2 & 3 & -2 & 1 \\ 9 & 15 & 1 & -1 \end{pmatrix}.$$

4.1.2 (a) The 2-level Haar transform array is

$$\begin{pmatrix} -4 & 1 & 0 & -1 \\ 5 & 9 & -3 & 1 \\ 9.5 & -2.5 & -2 & 1 \\ 14.5 & -3.5 & 1 & -1 \end{pmatrix}.$$

4.1.3 Since $\mathbf{V}_1^1$ and $\mathbf{W}_1^1$ have supports of length 2, it suffices to consider the case of a 2 by 2 image

$$\mathbf{f} = \begin{pmatrix} f_{1,2} & f_{2,2} \\ f_{1,1} & f_{2,1} \end{pmatrix}.$$

The 1-level Haar transform along rows yields

$$\begin{pmatrix} \frac{f_{1,2}+f_{2,2}}{\sqrt{2}} & \frac{f_{1,2}-f_{2,2}}{\sqrt{2}} \\ \\ \frac{f_{1,1}+f_{2,1}}{\sqrt{2}} & \frac{f_{1,1}-f_{2,1}}{\sqrt{2}} \end{pmatrix}$$

© 2008 by Taylor & Francis Group, LLC

and the 1-level Haar transforms along the columns yields

$$\begin{pmatrix} h_{1,1} & d_{1,1} \\ a_{1,1} & v_{1,1} \end{pmatrix} = \begin{pmatrix} \frac{f_{1,1}+f_{2,1}-f_{1,2}-f_{2,2}}{2} & \frac{f_{1,1}-f_{1,2}-f_{2,1}+f_{2,2}}{2} \\ \frac{f_{1,1}+f_{2,1}+f_{1,2}+f_{2,2}}{2} & \frac{f_{1,1}-f_{2,1}+f_{1,2}-f_{2,2}}{2} \end{pmatrix}. \tag{B.1}$$

Thus,

$$\begin{aligned} h_{1,1} &= \frac{f_{1,1}+f_{2,1}-f_{1,2}-f_{2,2}}{2} \\ &= \begin{pmatrix} f_{1,2} & f_{2,2} \\ f_{1,1} & f_{2,1} \end{pmatrix} \cdot \begin{pmatrix} -1/2 & -1/2 \\ 1/2 & 1/2 \end{pmatrix} \\ &= \mathbf{f} \cdot (\mathbf{V}_1^1 \otimes \mathbf{W}_1^1). \end{aligned}$$

4.1.5 We have

$$\mathbf{W}_1^1 \otimes \mathbf{V}_1^1 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 1/2 & -1/2 & 0 & \dots & 0 & 0 \\ 1/2 & -1/2 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

To prove that $v_{1,1} = \mathbf{f} \cdot (\mathbf{W}_1^1 \otimes \mathbf{V}_1^1)$, as in the solution to Exercise 4.1.3, it suffices to consider 2 by 2 images. Moreover, from Equation B.1 we obtain

$$\begin{aligned} v_{1,1} &= \frac{f_{1,1}-f_{2,1}+f_{1,2}-f_{2,2}}{2} \\ &= \begin{pmatrix} f_{1,2} & f_{2,2} \\ f_{1,1} & f_{2,1} \end{pmatrix} \cdot \begin{pmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{pmatrix} \\ &= \mathbf{f} \cdot (\mathbf{W}_1^1 \otimes \mathbf{V}_1^1). \end{aligned}$$

4.2.1 The PSNRs for the reconstructions of the `Airfield.bmp` image are

Method/C.R.	*8:1*	*16:1*	*32:1*
JPEG	29.3	26.6	23.5
J2K	31.5	28.3	25.5
ASWDR	31.4	28.4	25.7

4.3.2 The PSNRs for the zooms of the compressions are

Method/C.R.	*8:1*	*16:1*	*32:1*
JPEG	30.54	25.48	19.86
J2K	33.12	28.20	24.42
ASWDR	32.46	27.99	24.54

© 2008 by Taylor & Francis Group, LLC

4	6	8	6	2	4	2	6
−2	−2	−4	−2	0	2	−2	−4
−4	−4	−6	−4	2	−2	−6	−8
2	4	6	4	2	2	0	−2

6	4	6	8	10	−4	4	6
8	10	6	4	8	−6	6	8
14	28	−10	8	8	−6	4	10
22	24	−8	10	8	−4	8	8

(a) Threshold 2

5	7	9	7	3	5	3	7
−3	−3	−5	−3	1	3	−3	−5
−5	−5	−7	−5	3	−3	−7	−9
3	5	7	5	3	3	1	−3

7	5	7	9	11	−5	5	7
9	11	7	5	9	−7	7	9
15	29	−11	9	9	−7	5	11
23	25	−9	11	9	−5	9	9

(b) Half-threshold 1

FIGURE B.9
Quantized transforms for Exercise 4.4.1.

4.4.1 The quantized transform at threshold 2 is shown in Figure B.9(a), and for half-threshold 1 in (b).

4.4.5 + + + + + 1 1 1 0 1 +

4.4.11 A Huffman coding is

Symbol	*Encoding*
a	1
b	0 1
c	0 0 1
d	0 0 0

Its average length is

$$\frac{14}{21} \cdot 1 + \frac{5}{21} \cdot 2 + \frac{1}{21} \cdot 3 + \frac{1}{21} \cdot 3 = 1.42857\ldots$$

which is a bit larger than the entropy

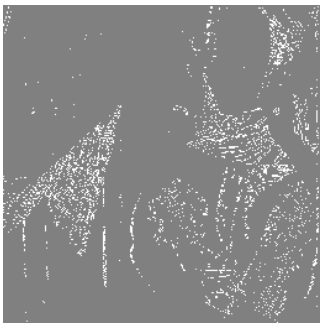
$$\frac{14}{21} \cdot \log_2\left(\frac{21}{14}\right) + \frac{5}{21} \cdot \log_2\left(\frac{21}{5}\right) + \frac{1}{21} \cdot \log_2(21) + \frac{1}{21} \cdot \log_2(21) = 1.30124\ldots$$

which is (essentially) obtainable with arithmetic coding.

4.5.1 The images and percentages of correct predictions are in Figure B.10. These data provide further confirmation of Principle A for wavelet transforms of images.

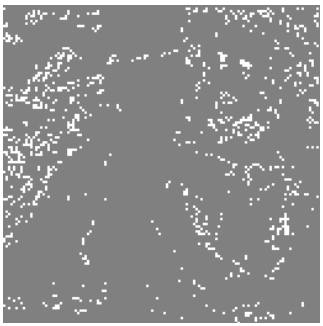
4.6.1 The images illustrating progressive reconstruction are in Figure B.11. As pointed out in the text, the reconstructions shown there were all produced from one

© 2008 by Taylor & Francis Group, LLC

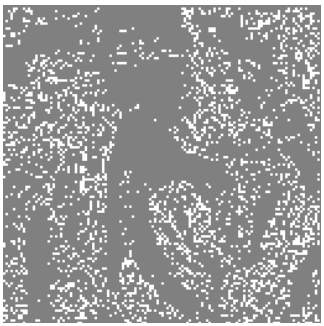


(a) $\mathbf{v}_2$ predictions for $\mathbf{v}_1$

(b) $\mathbf{v}_1$ new significant values



(c) $\mathbf{h}_3$ predictions for $\mathbf{h}_2$



(d) $\mathbf{h}_2$ new significant values

FIGURE B.10
Illustration of Parent-Child significance correlation for Barb image. (a) and (c) are predictions of locations (white pixels) of new significant values based on parent values significant at threshold 16**. (b) and (d) are actual locations (white pixels) of *new* significant values at child positions for threshold** 8**. For (a) and (b),** 49.7% **of the predictions were correct. For (c) and (d),** 39.5% **of the predictions were correct.**

file (compressed to 1.0 bpp). This example illustrates that *from that one file,* images at any desired compression rate (up to 1.0 bpp) can be reconstructed.

4.6.5 The images are shown in Figure B.12. If the entire image is transmitted losslessly, then 187, 044 bytes are needed. If only the image in (d) is sent (with just the ROI lossless), then only 10, 264 bytes are needed, a 94.5% savings.

4.6.7 The results are

Image/Method	WinZip®	WinRAR®	ASWDR
Barbara	10%	20%	39%
Boats	23%	33%	45%
X-ray	45%	61%	68%

4.7.3 The RD-curves are shown in Figure B.13. J2K performs slightly better than

© 2008 by Taylor & Francis Group, LLC

(a) Original image

(b) 0.25 bpp (32:1 compression)

(c) 0.5 bpp (16:1 compression)

(d) 1.0 bpp (8:1 compression)

FIGURE B.11
Illustration of Progressive Reconstruction. Each image in (b) to (d) was computed from a single compressed file (saved at 1.0 bpp). First, (b) is reconstructed, then (c), and then (d).

ASWDR for this image.

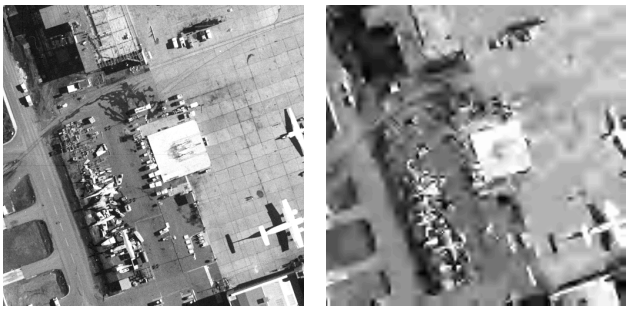
4.8.5 The denoisings are shown in Figure B.14. Their subjective interpretation is left to the reader.

4.9.2 The edge enhancement is shown in Figure B.15. As in the text, we computed a 1-level Daub4 wavelet transform, multiplied its fluctuation values by 3, and then inverse transformed. The resulting image was then quantized at 8 bpp.

4.9.6 The relative 2-norm differences for these images (compared to a 32:1 decompression of `Lena.pgm`) are

Image	*Full*	*Second*	*Third*	*Fourth*	*Fifth*
Barb	0.499	0.467	0.455	0.433	0.379
Zelda	0.548	0.543	0.537	0.524	0.490
Lena	0.040	0.015	0.009	0.005	0.002
Barb	0.609	0.604	0.596	0.579	0.537

© 2008 by Taylor & Francis Group, LLC



(a) Original image (b) 200:1 compression

(c) Original with ROI (d) Update of (b), lossless ROI

FIGURE B.12
Illustration of ROI Property.

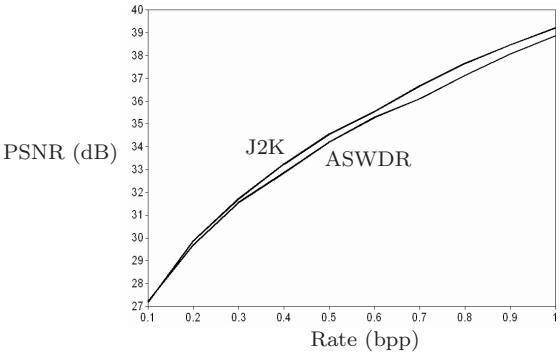


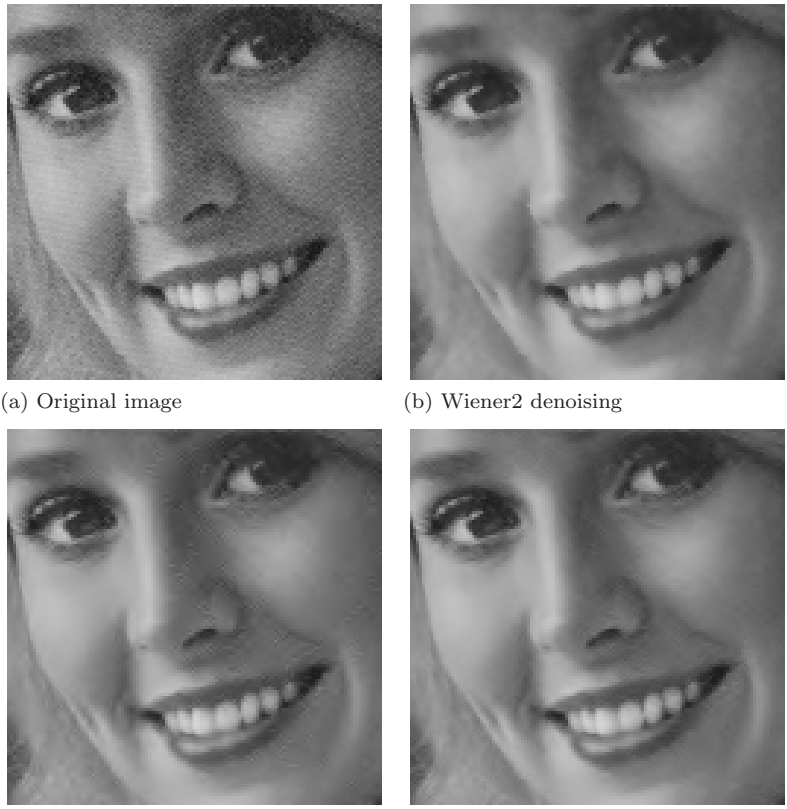
FIGURE B.13
Rate-Distortion curves for the `Boat.bmp` image.

B.5 Chapter 5

5.1.1 The graphs are shown in Figure B.16.

5.1.3 The spectra are shown in Figure B.17.

© 2008 by Taylor & Francis Group, LLC



(a) Original image (b) Wiener2 denoising

(c) TAWS denoising (d) TAWS-SPIN denoising

FIGURE B.14
Zooms of denoisings of `Elaine.bmp` image. Their subjective interpretation is left to the reader.

5.2.1 The n^{th} value of the DFT of $\alpha\mathbf{f} + \beta\mathbf{g}$ is

$$\sum_{m=1}^{N} (\alpha f_m + \beta g_m) e^{-i2\pi(m-1)(n-1)/N}$$

$$= \alpha \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} + \beta \sum_{m=1}^{N} g_m e^{-i2\pi(m-1)(n-1)/N}$$

$$= \alpha(\mathcal{F}\mathbf{f})_n + \beta(\mathcal{F}\mathbf{g})_n,$$

which shows that the DFT of $\alpha\mathbf{f} + \beta\mathbf{g}$ is $\alpha\mathcal{F}\mathbf{f} + \beta\mathcal{F}\mathbf{g}$.

© 2008 by Taylor & Francis Group, LLC

(a) Original (b) Edge enhancement

FIGURE B.15
Edge enhancement for Exercise 4.9.2.

5.2.2 We have for each $n = 1, 2, \ldots, N$:

$$(\mathcal{F}\mathbf{f})_{n+N} = \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1+N)/N}$$

$$= \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} e^{-i2\pi(m-1)}$$

$$= \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} \cdot 1$$

$$= (\mathcal{F}\mathbf{f})_n$$

and that proves the periodicity.

© 2008 by Taylor & Francis Group, LLC

5.2.3 Fix a value of m from 1 to N. We then have

$$\frac{1}{N}\sum_{n=-N/2}^{N/2-1}(\mathcal{F}\mathbf{f})_n\, e^{+i2\pi(n-1)(m-1)/N}$$

$$= \sum_{n=-N/2}^{N/2-1}\frac{1}{N}\sum_{\ell=1}^{N} f_\ell e^{-i2\pi(\ell-1)(n-1)/N}\, e^{+i2\pi(n-1)(m-1)/N}$$

$$= \sum_{\ell=1}^{N} f_\ell \left(\frac{1}{N}\sum_{n=-N/2}^{N/2-1} e^{-i2\pi(\ell-1)(n-1)/N}\, e^{+i2\pi(n-1)(m-1)/N}\right)$$

$$= \sum_{\ell=1}^{N} f_\ell \left(\frac{1}{N}\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N}\right).$$

We now simplify the innermost sum in the last line above. If $\ell = m$, then we have

$$\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N} = \sum_{n=-N/2}^{N/2-1} e^{0} = N.$$

If $\ell \neq m$, then we have (using a finite geometric series sum):

$$\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N} = \sum_{n=-N/2}^{N/2-1} (e^{i2\pi(m-\ell)/N})^{n-1}$$

$$= \frac{e^{-i\pi(m-\ell)}e^{-i2\pi(m-\ell)/N} - e^{i\pi(m-\ell)}e^{-i2\pi(m-\ell)/N}}{1 - e^{i2\pi(m-\ell)/N}}$$

$$= \frac{(e^{-i\pi(m-\ell)} - e^{i\pi(m-\ell)})e^{-i2\pi(m-\ell)/N}}{1 - e^{i2\pi(m-\ell)/N}}$$

$$= 0$$

since $e^{\pm i\pi(m-\ell)}$ have the same value (of either -1 or $+1$). Therefore, we have

$$\frac{1}{N}\sum_{n=-N/2}^{N/2-1}(\mathcal{F}\mathbf{f})_n e^{+i2\pi(m-1)(n-1)/N} = f_m \cdot 1$$

and inversion is proved.

5.2.5 The z-transform is $1 + z + z^2 + z^3$ and its roots are $-1, \pm i$.

© 2008 by Taylor & Francis Group, LLC

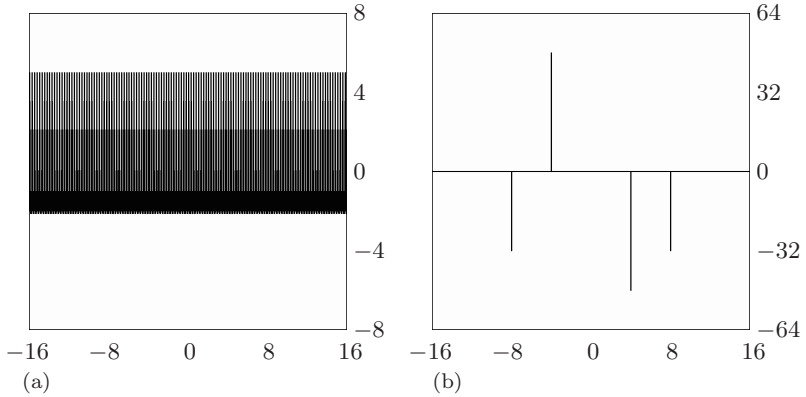


FIGURE B.16
(a) Graph of function. (b) Frequency content.

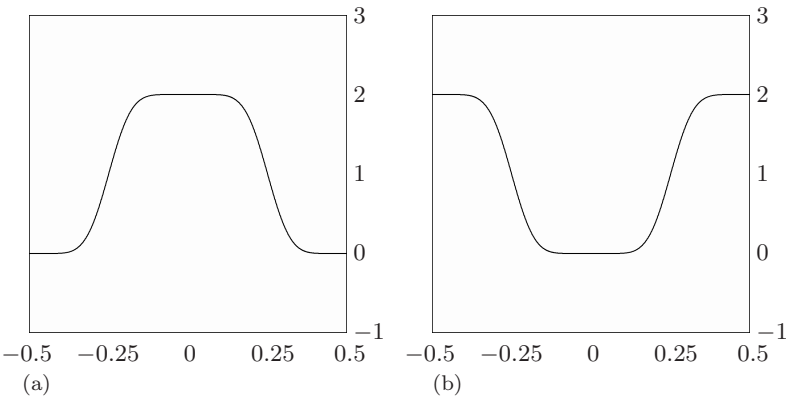


FIGURE B.17
(a) Spectrum of Coif12 scaling signal $\mathbf{V}_{20}^1$. (b) Spectrum of Coif12 wavelet $\mathbf{W}_{20}^1$.

5.3.1 The portions of the DFT that correspond to $\mathbf{A}^3$, $\mathbf{D}^3$, $\mathbf{D}^2$, and $\mathbf{D}^1$ are shown in Figure B.18.

5.5.3 Before doing normalized correlation, both the text image and the single letter image are *inverted* in the sense that black pixels are converted to white pixels and vice versa. As to the relation to vision, we know that there are neurons that respond to darkness in the foreground versus brightness in the background, just as there are neurons that respond to whiteness in the foreground versus darkness in the background. These complementary responses of neurons are loosely analogous to how we can choose to invert the black and white relationships for our normalized correlations.

5.6.3 To see the solution, you should consult the article by Strichartz (reference [18] for Chapter 5) where the Daub6 case is discussed in detail.

© 2008 by Taylor & Francis Group, LLC

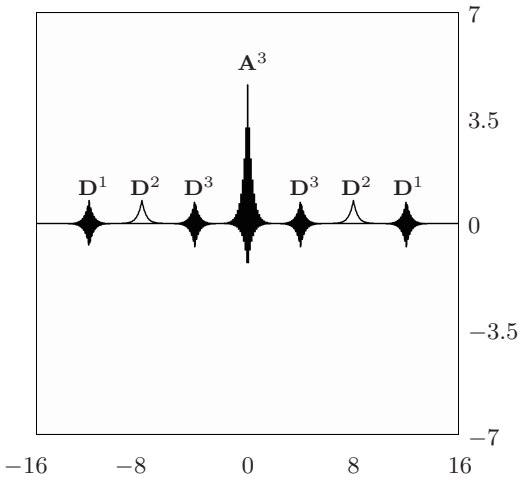
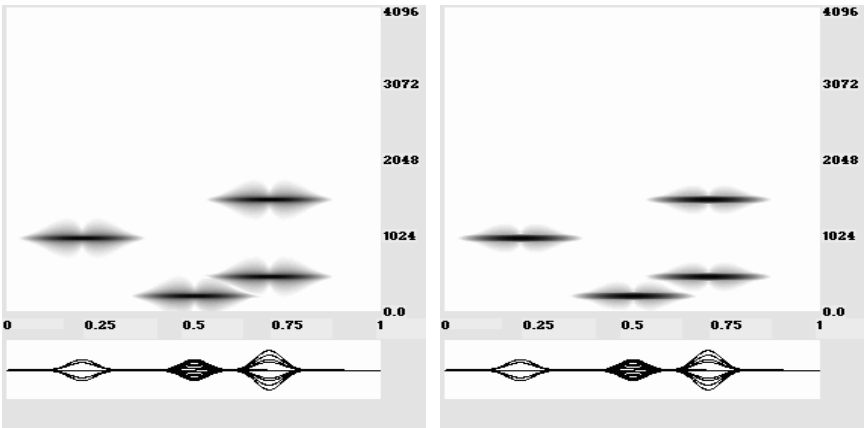


FIGURE B.18
DFT with parts labelled (e.g., A^3 labels the DFT portion corresponding to A^3).



(a) (b)

FIGURE B.19
(a) Hanning windowed spectrogram. (b) Blackman windowed spectrogram.

5.7.1 The spectrograms are shown in Figure B.19. They are quite similar. The Hanning windowed spectrogram, however, shows a bit more "smearing" (slightly more extensive light gray patches above and below the darker frequency bands). Engineers refer to this smearing as "leakage." Because they show less leakage, we generally use Blackman windows for our spectrograms.

5.8.2^c The spectrogram is shown in Figure B.20. The two diagonal segments—forming a top of a triangle shaped object enclosed in the rectangle at bottom left center—is repeated with different sizes and positions in the time-frequency plane,

© 2008 by Taylor & Francis Group, LLC

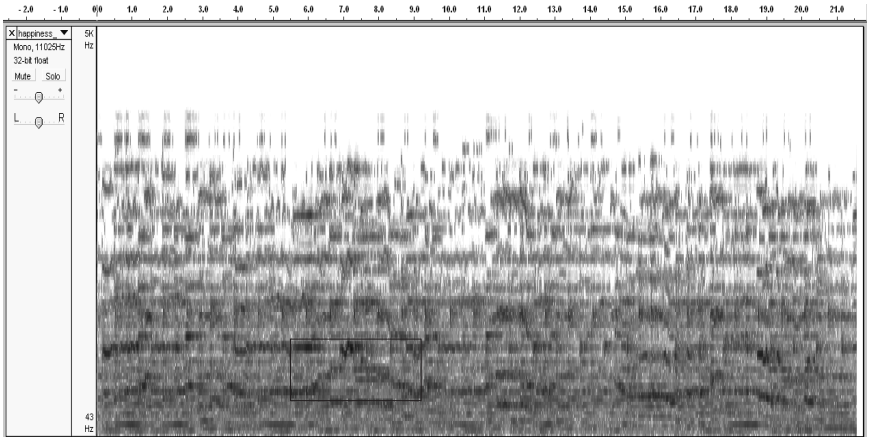


FIGURE B.20
AUDACITY computed spectrogram of `Happiness_clip.wav`.

and is also inverted in a region at its right edge and again inverted (over a slightly longer time-scale) at the end of the spectrogram. All of these structures are even clearer in the color version produced by AUDACITY, and reveal their full effect when the spectrogram is traced as the clip is played. This analysis provides further confirmation of our Multiresolution Principle.

5.9.2 A musical analysis of the `warbler.wav` recording can be found in subsection **2.3 Analysis III: A Warbler's Song** of the article *Music: a time-frequency approach* available at

`http://www.uwec.edu/walkerjs/media/TFAM.pdf` (B.2)

5.9.4 One solution to selectively amplifying the harp glissando can be found in subsection **3.3 Synthesis II: Altering figure-ground** of the article *Music: a time-frequency approach* available at the webpage listed in (B.2). More details on how FAWAV can be used for this example can be found in the document listed in the statement of problem 5.8.2 on p. 217.

B.6 Chapter 6

6.1.1 $(10, 10\,|\,-4, -4\,|\,0, -2\,|\,-2, 0)$

6.1.3 Wavelet packet series: 183. Wavelet series: 228.

6.2.1 Using the *Threshold* choice for a wavelet and wavelet packet series, with threshold setting `1/2^7:1/2^5` in both cases, we obtained the following results:

© 2008 by Taylor & Francis Group, LLC

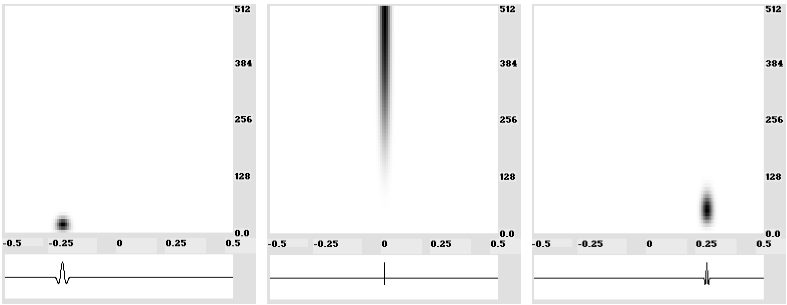


FIGURE B.21
Spectrograms for Problem 6.3.1.

Transform	*Sig. values*	*Bpp*	*RMS Error*
wavelet	1219	0.26	189.5
wavelet packet	1185	0.25	182.2

There are some high-frequency artifacts audible in the compressions. These are very high compressions ($\approx$ 64:1); a lower compression ratio (using lower threshold settings) would undoubtedly sound better.

6.2.4 The RMS error between the original and the noisy signal is 1003.6. We applied a 5-level Coif30 transform and used the thresholding `g1(x)(abs(g1(x))>4000)` (the threshold of `4000` was obtained by visual inspection of the transform). After inverse transforming the thresholded transform, we obtained a denoised signal with RMS error of 969.1, but there was much less audible noise. Performing the same thresholding on a 5-level Coif30 wavelet packet transform, we obtained a denoised signal with RMS error of 811.6, which is smaller, and this denoising also sounded better.

The wavelet packet denoising was better because there is a lot of energy within the fluctuations of the osprey's call. By performing wavelet transforms on those fluctuations we amplify the signal dominated fluctuations, while at the same time leaving the noise variance unchanged. That allows for a more effective thresholding operation.

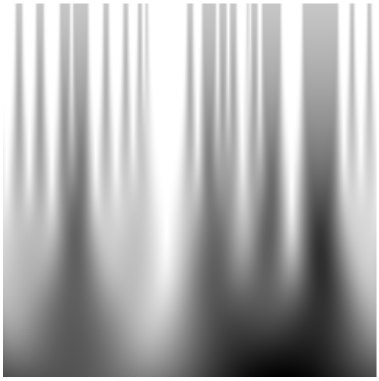
6.2.6 The analog of Table 6.2 for the `Boat.bmp` image is the following:

Transform	*Bpp*	*Sig. values*	*PSNR*
wavelet	0.5	24034	34.12
wavelet packet	0.5	22229	33.17
wavelet	0.25	11557	30.59
wavelet packet	0.25	10646	29.94

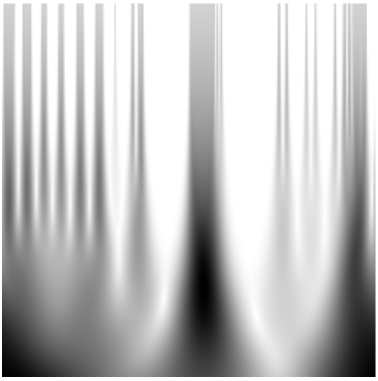
6.3.1 The spectrograms are shown in Figure B.21.

6.5.1 to 6.5.4 The percussion scalograms are shown in Figure B.22. We used the default values assigned by FAWAV for the parameters in each percussion scalogram (see reference [22] in Chapter 6 for an explanation of how these parameter values for

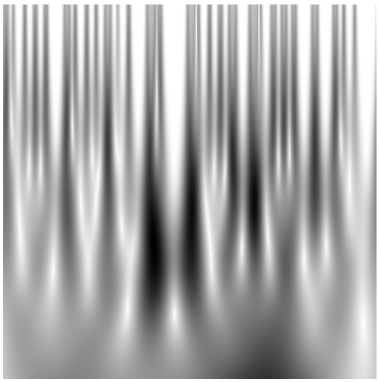
© 2008 by Taylor & Francis Group, LLC



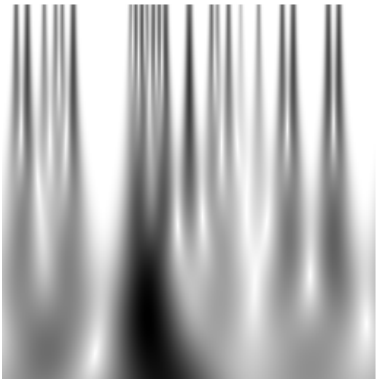
(a) 6.5.1: Mingus_Dizzy_(Clip_1).wav



(b) 6.5.2: Mingus_Dizzy_(Clip_2).wav



(c) 6.5.3: Slapshift_(Clip_1).wav



(d) 6.5.4: Slapshift_(Clip_2).wav

FIGURE B.22
Percussion scalograms for Exercises 6.5.1 to 6.5.4.

percussion scalograms are chosen). For graphs (a) to (c), a percussion scalogram was computed with no processing done on the Blackman windowed spectrogram. For (d), the graph of `g1(y>1500)` was computed for the spectrogram before selecting the percussion scalogram procedure.

In each case it is easy to pick out the individual notes (or percussive strikes) and their grouping into hierarchies.

© 2008 by Taylor & Francis Group, LLC

Appendix C

Wavelet software

A computational study is unlikely to lead to real scientific progress unless the software environment is convenient enough to encourage one to vary parameters, modify the problem, play around.

—Lloyd N. Trefethen

I don't think I could have done anything if I hadn't had a little computer and some graphics output.

—Alex Grossmann[1]

In order to apply wavelets to practical problems, a computer software environment is essential. The software FAWAV which produced the figures for this primer provides such an environment. In addition to producing the examples for the primer, I have also used it for producing most of the examples for my own research in wavelet and time-frequency analysis. FAWAV can be downloaded from the FAWAV website:

http://www.uwec.edu/walkerjs/FAWAVE/

free of charge. When you visit the FAWAV website there will be instructions for how to proceed (see also Section A.1 below). There is much more material available at the book website:

http://www.uwec.edu/walkerjs/Primer/

[1] Trefethen's quote is from [1]. Grossmann's quote is from [2].

© 2008 by Taylor & Francis Group, LLC

including: (1) a comprehensive introduction to the use of FAWAV for digital signal processing in its *User Manual;* (2) additional sounds and images for use with FAWAV; (3) suggestions for projects, and sample projects, using FAWAV in the *Project Ideas* section of the website; (4) links to other wavelet material; (5) an online bibliography with links to downloadable papers for further study.

C.1 Installing the book's software

To install FAWAV on your computer you go to the FAWAV website and click on the `Download Fawave (Version 2)` link. The setup program is a standard WINDOWS® setup program, so we shall not describe it any further, except to say that as a safeguard it will not overwrite any existing files on your computer. Later, if you want to uninstall FAWAV, you can do so by selecting the *Add/Remove Programs* option from the WINDOWS® *Control Panel.*

Note: Although FAWAV was designed in a WINDOWS® environment, my students have told me that it does install and run adequately under both MACINTOSH® and LINUX operating systems, when using programs designed to run WINDOWS® applications on those systems (such as the WINE program for LINUX).

To speed up downloading, this installation of FAWAV does *not* copy all of the data files—such as image files and audio files—to your computer. You should download these files separately from either the FAWAV or book websites as you need them.

C.2 Other software

FAWAV can only serve as an introductory tool for applications of wavelet analysis. It does not contain all of the huge number of various algorithms that have been created, nor does it include all of the manifold types of wavelets that exist. Fortunately, there are several excellent software packages available, many of them freely available for downloading from the Internet. Here is a partial list:

- WAVELAB is a set of MATLAB® routines developed by Donoho and his collaborators at Stanford. It can be downloaded from the following website:

  http://www-stat.stanford.edu/∼wavelab

- WAVELET TOOLS is a collection of MATLAB® routines written by researchers at Rice University. It is available from the following website:

  http://www-dsp.rice.edu/software/RWT

- LASTWAVE is a set of C language routines, including a command shell environment written by several French researchers. It can be obtained

© 2008 by Taylor & Francis Group, LLC

from the website:

http://www.cmap.polytechnique.fr/users/www.bacry

- AWA3 (Adapted Waveform Analysis Library, v. 3) is a collection of C language routines for wavelet analysis produced by *Fast Mathematical Algorithms and Hardware Corporation.* For further information, send an e-mail to the following address:

victor@math.wustl.edu

There are many other software packages available. The book [3] contains a listing of wavelet analysis software, as do [4] and [5]. A great deal of the latest information on wavelet analysis—including announcements of new software and the latest updates to existing software—can be obtained by signing on to the free electronic journal, *Wavelet Digest,* at the website [6].

C.3 References

1. L.N. Trefethen. (1998). Maxims about numerical mathematics, computers, science, and life. *SIAM News,* Vol. 31, No. 1.
2. B. Burke. (1994). The mathematical microscope: waves, wavelets, and beyond. In *A Positron Named Priscilla, Scientific Discovery at the Frontier,* 196–235, Nat. Academy Press. Edited by M. Bartusiak.
3. B. Burke Hubbard. (1998). *The World According to Wavelets, Second Edition.* AK Peters, Wellesley, MA.
4. S. Mallat. (1999). *A Wavelet Tour of Signal Processing. Second Edition.* Academic Press, New York, NY.
5. C.S. Burrus, R.H. Gopinath, and H. Guo. (1998). *Introduction to Wavelets and Wavelet Transforms, A Primer.* Prentice Hall, Englewood Cliffs, NJ.
6. The *Wavelet Digest* website: http://www.wavelet.org

© 2008 by Taylor & Francis Group, LLC

Bibliography

P.S. Addison. (2005). Wavelet transforms and the ECG: a review. *Physiol. Meas.,* Vol. 26, R155–R199.

M. Akay. (1996). Diagnosis of coronary artery disease using wavelet-based neural networks. In A. Aldroubi and M. Unser, Eds., *Wavelets in Medicine and Biology.* CRC Press, Boca Raton, FL, 513–526.

A. Aldroubi and M. Unser, Eds. (1996). *Wavelets in Medicine and Biology.* CRC Press, Boca Raton, FL.

J.F. Alm and J.S. Walker. (2002). Time-frequency analysis of musical instruments. *SIAM Review,* Vol. 44, 457–476.

R.B. Ash. (1990). *Information Theory.* Dover, New York, NY.

G. Assayag, H. Feichtinger, and J.F. Rodrigues, Eds. (2002). *Mathematics and Music: a Diderot Mathematical Forum.* Springer, New York, NY

`Audacity` software. Available at `http://audacity.sourceforge.net/`

Avian music website: `http://www.avianmusic.com/music.html/`

M. Babbitt. (1961). Set structure as a compositional determinant. *J. of Music Theory,* Vol. 5, 72–94.

S.I. Baskakov. (1986). *Signals and Circuits.* Mir, Moscow.

T.C. Bell, J.G. Cleary, and I.H. Witten. (1990). *Text Compression.* Prentice Hall, Englewood Cliffs, NJ.

D. Benson. (2006). *Music: A Mathematical Offering.* Cambridge University Press, Cambridge, UK.

Book's website: `http://www.uwec.edu/walkerjs/Primer`

J.N. Bradley, C.M. Brislawn, and T. Hopper. (1993). The FBI Wavelet/Scalar Quantization Standard for gray-scale fingerprint image compression. *SPIE,* Vol. 1961, *Visual Information Processing II (1993)*, 293–304.

© 2008 by Taylor & Francis Group, LLC

O. Bratteli and P. Jorgensen. (2002). *Wavelets through a Looking Glass.* Birkhäuser, Boston, MA.

W.L. Briggs and V.E. Henson. (1995). *The DFT. An Owner's Manual.* SIAM, Philadelphia, PA.

E.O. Brigham. (1978). *The Fast Fourier Transform.* Prentice Hall, Englewood Cliffs, NJ.

C.M. Brislawn. (1995). Fingerprints go digital. *Notices of the Amer. Math. Soc.,* Vol. 42, 1278–1283.

B. Burke. (1994). The mathematical microscope: waves, wavelets, and beyond. In *A Positron Named Priscilla, Scientific Discovery at the Frontier,* 196–235, Nat. Academy Press. Edited by M. Bartusiak.

C.S. Burrus, R.H. Gopinath, and H. Guo. (1998). *Introduction to Wavelets and Wavelet Transforms, A Primer.* Prentice Hall, Englewood Cliffs, NJ.

Calderbank, A. R., et al. (1998). Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis,* Vol. 5, 332–369.

P.G. Casazza. (2000). The art of frame theory. *Taiwanese J. of Mathematics,* Vol. 4, 129–201.

Y.-J. Chen, S. Oraintara, and K. Amaratunga. (2005). Dyadic-based factorization for regular paraunitary filter banks and M-band orthogonal wavelets with structural vanishing moments. *IEEE Transactions on Signal Processing,* Vol. 53, 193–207.

X. Cheng, J.V. Hart, and J.S. Walker. (2007). Time-frequency analysis of musical rhythm. Preprint. Available at http://www.uwec.edu/walkerjs/media/TFAMR.pdf

H.A. Chipman, E.D. Kolaczyk, and R.E. McCulloch. (1997). Adaptive Bayesian wavelet shrinkage. *J. American Statistical Association,* Vol. 92, 1413–1421.

O. Christiansen. (2002). Time-frequency analysis and its applications in denoising. Thesis, Department of Informatics, University of Bergen.

C.K. Chui (Ed.). (1992). *Wavelets: A Tutorial in Theory and Applications.* Academic Press, Boston, MA.

C.K. Chui. (1997). *Wavelets: A Mathematical Tool for Signal Analysis.* SIAM, Philadelphia, PA.

E. Coen. (1999). *The Art of Genes.* Oxford University Press, Oxford, UK.

R. Cogan. (1984). *New Images of Musical Sound.* Harvard University Press, Cambridge, MA.

Cohen, A., Daubechies, I., and Feauveau, J.-C. (1992). Biorthogonal bases of compactly supported wavelets. *Commun. on Pure and Appl. Math.,* Vol. 45, 485–560.

R. Coifman and D. Donoho. (1995). Translation-invariant denoising. In *Wavelets and Statistics,* A. Antoniadis and G. Oppenheim, Eds. Springer, New York, NY.

R.R. Coifman and M.V. Wickerhauser. (1993). Wavelets and adapted waveform analysis. A toolkit for signal processing and numerical analysis. In *Different Perspectives on Wavelets,* I. Daubechies (Ed.), AMS, Providence, RI, 119–154.

R.R. Coifman and M.V. Wickerhauser. (1994). Wavelets and adapted waveform analysis. In *Wavelets. Mathematics and Applications,* J. Benedetto and M. Frazier (Eds.), CRC Press, Boca Raton, FL, 399–424.

© 2008 by Taylor & Francis Group, LLC

R. Coifman and Y. Zeevi, Eds. (1988). *Signal and Image Representation in Combined Spaces.* Wavelet Analysis and Applications, Vol. 7, Academic Press, Boston, MA.

T.M. Cover, J.A. Thomas. (1991). *Elements of Information Theory.* Wiley, New York, NY.

Curvelet website: http://www.curvelet.org/papers.html

I. Daubechies. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA.

I. Daubechies and S. Maes. (1996). A nonlinear squeezing of the continuous wavelet transform based on auditory nerve models. In *Wavelets in Medicine and Biology,* A. Aldroubi and M. Unser (Eds.), CRC Press, Boca Raton, FL, 527–546.

I. Daubechies and W. Sweldens. (1998). Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.,* Vol. 4, 247–269.

G.M. Davis. (1998). A wavelet-based analysis of fractal image compression. *IEEE Trans. on Image Proc.,* Vol. 7, 141–154.

G.M. Davis and A. Nosratinia. (1998). Wavelet-based image coding: an overview. *Applied and Computational Control, Signals and Circuits,* Vol. 1, 205–269.

D. Donoho. (1993). Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. In I. Daubechies (Ed.), *Different Perspectives on Wavelets.* AMS, Providence, RI, 173–205.

D. Donoho and E. Candes. (2005). Continuous curvelet transform I: resolution of wavefront set. *Applied and Computational Harmonic Analysis,* Vol. 19, 162–197.

D. Donoho and E. Candes. (2005). Continuous curvelet transform II: discretization and frames. *Appl. Comput. Harmon. Anal.* Vol. 19, 198–222.

D. Donoho, et al. (1995). Wavelet shrinkage: asymptopia? *J. Royal Stat. Soc. B,* Vol. 57, 301–369.

D.L. Donoho and A.G. Flesia. (2001). Can recent innovations in harmonic analysis 'explain' key findings in natural image statistics? *Network Computations in Neural Systems,* Vol. 12, 371–393.

D. Donoho and I. Johnstone. (1994). Ideal spatial adaptation via wavelet shrinkage. *Biometrika,* Vol. 81, 425–455.

D. Donoho and I. Johnstone. (1995). Adapting to unknown smoothness via wavelet shrinkage. *American Statistical Assoc.,* Vol. 90, 1200–1224.

M. Dörfler. (2002). Gabor Analysis for a Class of Signals called Music. Dissertation, University of Vienna.

M. Dörfler, H. Feichtinger. (2004). Quilted Gabor families I: reduced multi-Gabor frames. *Appl. Comput. Harmon. Anal.,* Vol. 356, 2001–2023. Available at

http://www.mat.univie.ac.at/~moni/

R.L. Fante. (1988). *Signal Analysis and Estimation.* Wiley, New York, NY.

H. Feichtinger and T. Strohmer, Eds. (1998). *Gabor Analysis and Algorithms.* Birkhäuser, Boston, MA.

H. Feichtinger and T. Strohmer, Eds. (2002). *Advances in Gabor Analysis.* Birkhäuser, Boston, MA.

© 2008 by Taylor & Francis Group, LLC

D. Field. (1993). Scale invariance and self-similar "wavelet" transforms: an analysis of natural scenes and mammalian visual systems. In *Wavelets, Fractals and Fourier Transforms,* M. Farge, J. Hunt, and J. Vassilicos, Eds. Clarendon Press, Oxford, UK, 151–193.

D. Field. (1994). What is the goal of sensory coding? *Neural Computations,* Vol. 6, 559–601.

D. Field. (1999). Wavelets, vision and the statistics of natural scenes. *Phil. Trans. R. Soc. London A,* Vol. 357, 2527–2542.

M. Frazier. (2001). *An Introduction to Wavelets Through Linear Algebra.* Springer, New York, NY.

H.-Y. Gao. (1998). Wavelet shrinkage denoising using the non-negative garrote. *J. of Computational and Graphical Statistics,* Vol. 7, 469–488.

T. Gardner and M. Magnasco. (2006). Sparse time-frequency representations. *Proc. Nat. Acad. Sci.,* Vol. 103, 6094–6099. Available at

http://www.pnas.org/cgi/reprint/103/16/6094.pdf

R. Gregory, Ed. (2004). *The Oxford Companion to the Mind,* Oxford University Press, Oxford, UK.

K. Gröchenig. (2001). *Foundations of Time-Frequency Analysis.* Birkhäuser, Boston, MA.

I. Guskov, W. Sweldens, and P. Schröder. (1999). Multiresolution signal processing for meshes. *SIGGRAPH 1999,* 325–334.

R.W. Hamming. (1987). *Numerical Methods for Scientists and Engineers.* Dover, New York, NY.

D.K. Hammond and E.P. Simoncelli. (2006). Image denoising with an orientation-adaptive Gaussian scale mixture model. *Proc. 13th IEEE Int'l Conference on Image Processing,* Atlanta, Georgia.

D. Hankerson, G.A. Harris, and P.D. Johnson, Jr. (2003). *Introduction to Information Theory and Data Compression, Second Edition.* CRC Press, Boca Raton, FL.

L. Harkleroad. (2006). *The Math Behind the Music.* Cambridge University Press, Cambridge, UK.

D.J. Heeger, E.P. Simoncelli, and J.A. Movshon. (1996). Computational models of cortical visual processing. *Proc. National Academy of Science,* Vol. 93, 623–627.

V.K. Heer and H-E Reinfelder. (1990). Comparison of reversible methods for data compression. In *Medical Imaging IV,* 354–365. Proceedings SPIE, No. 1233.

E. Hernandez and G. Weiss. (1996). *A First Course on Wavelets.* CRC Press, Boca Raton, FL.

W. Hodges and R. Wilson. (2002). Musical patterns. In G. Assayag, H.G. Feichtinger, and J.F. Rodrigues, Eds., *Mathematics and Music.* Springer-Verlag, NY.

B. Burke Hubbard. (1998). *The World According to Wavelets, Second Edition.* AK Peters, Wellesley, MA.

Image archive: www.uwec.edu/walkerjs/Primer/Data/Primer_Images.zip

© 2008 by Taylor & Francis Group, LLC

Image quality indices webpage: Available at http://www.cns.nyu.edu/~zwang/

P. Isihara and M. Knapp. (1993). Basic Z_{12} analysis of musical chords. *UMAP Journal,* Vol. 14, 319–348.

B. Jähne. (2002). *Digital Image Processing, 5th Ed.* Springer, New York, NY.

B.D. Johnson. (2002). *Wavelets: generalized quasi-affine and oversampled-affine frames.* Thesis, Washington University in St. Louis.

A. Khodakovsky, P. Schröder, and W. Sweldens. (2000). Progressive Geometry Compression. *SIGGRAPH 2000,* 271–278.

M. Kobayashi. (1996). Listening for defects: wavelet-based acoustical signal processing in Japan. *SIAM News,* Vol. 29, No. 2.

D. Kroodsma. (2005). *The Singing Life of Birds.* Houghton-Mifflin, NY.

D. Legall and A. Tabatai. (1988). Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *Int. Conf. Acoustic, Speech, Signal Processing,* New York, 761–765.

F. Lerdahl and R. Jackendoff. (1983). *A Generative Theory of Tonal Music.* MIT Press, Cambridge, MA.

F. Lerdahl and R. Jackendoff. (1992). An overview of hierarchical structure in music. In *Machine Models of Music,* S. Schwanauer and D. Levitt, Eds., MIT Press, Cambridge, MA, 289–312.

J. Li. (2002). Image Compression—The mechanics of the JPEG 2000. Available at

research.microsoft.com/~jinl/paper_2002/msri_jpeg.htm

H. Longuet-Higgins. (1994). Artificial intelligence and musical cognition. *Phil. Trans. Roy. Soc. London,* Vol. A349, 115–131.

A.K. Louis, P. Maaß, and A. Rieder. (1997). *Wavelets, Theory and Applications.* Wiley, New York, NY.

G. Loy. (2007). *Musimathics: The Mathematical Foundations of Music, Vol. 2.* MIT Press, Cambridge, MA.

F. Luisier, T. Blu, and M. Unser. (2007). A new SURE approach to image denoising: inter-scale orthonormal wavelet thresholding. *IEEE Trans. Image Processing,* Vol. 16, 593–606.

S. Lyu and E.P. Simoncelli. (2007). Statistical modeling of images with fields of Gaussian scale mixtures. In *Adv. Neural Information Processing Systems,* Vol. 19.

F.B. Mache. (1993). *Music, Myth and Nature. Contemporary Music Studies, Vol. 6.* Taylor & Francis, London.

S. Mallat. (1999). *A Wavelet Tour of Signal Processing. Second Edition.* Academic Press, New York, NY.

H.S. Malvar and D.H. Staelin. (1989). The LOT: transform coding without blocking effects. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. 37, 553–559.

G. Mazzola. (2002). *The Topos of Music.* Birkhäuser, Basel.

© 2008 by Taylor & Francis Group, LLC

MetaSynth music software, available at:

http://www.uisoftware.com/MetaSynth/

Y. Meyer. (1993). *Wavelets. Algorithms and Applications.* SIAM, Philadelphia, PA.

M. Nelson. (1991). Arithmetic Coding + Statistical Modeling = Data Compression. Available at http://www.dogma.net/markn/articles/arith/part1.htm

A. Nestke and T. Noll. (2000). Inner Metrical Analysis. In J. Haluska (Ed.), *Music and Mathematics,* Tatra Mountains Mathematical Publications, Bratislava.

T.Q. Nguyen. (1992). A class of generalized cosine-modulated filter bank. *Proceedings 1992 IEEE International Symposium on Circuits and Systems,* Vol. 2, 943–946.

NuHAG webpage: http://www.univie.ac.at/nuhag-php/home/

W. O'Grady, M. Dobrovolsky, and M. Arnoff. (1993). *Contemporary Linguistics, An Introduction.* St. Martins Press, New York.

S. Oraintara, P. Heller, T. Tran, and T. Nguyen. (2001). Lattice structure for regular paraunitary linear-phase filterbanks and M-band orthogonal symmetric wavelets. *IEEE Transactions on Signal Processing,* Vol. 49, 2659-2672.

S. Pinker. (1997). *How the Mind Works.* Norton, NY.

J. Portilla, V. Strela, M. Wainwright, and E.P. Simoncelli. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Processing,* Vol. 12, 1338–1351.

R.L. de Queiroz, T.Q. Nguyen, and K.R. Rao. (1996). The GenLOT: generalized linear-phase lapped orthogonal transform. *IEEE Trans. on Signal Processing,* Vol. 44, 497–507.

M. Rabbini and P.W. Jones. (1991). *Digital Image Compression Techniques.* SPIE Press, Bellingham, WA.

K.R. Rao. (1982). *Fast Transforms: Algorithms, Analyses, Applications.* Academic Press, New York, NY.

H.L. Resnikoff and R.O. Wells, Jr. (1998). *Wavelet Analysis. The Scalable Structure of Information.* Springer, New York, NY.

D. Rothenberg. (2005). *Why Birds Sing: A Journey into the Mystery of Bird Song.* Basic Books, NY.

J.C. Russ. (1995). *The Image Processing Handbook.* CRC Press, Boca Raton, FL.

A. Said. (2004). *Introduction to Arithmetic Coding – Theory and Practice.* Chap. 5 of *Lossless Compression Handbook* (2003), K. Sayood (Ed.), Academic Press, San Diego, 101–152. Available at

http://www.hpl.hp.com/techreports/2004/HPL-2004-76.pdf

S. Schwanauer and D. Levitt, Eds. (1993). *Machine Models of Music.* MIT Press, Cambridge, MA.

L. Senhadji, L. Thoraval, and G. Carrault. (1996). Continuous wavelet transform: ECG recognition based on phase and modulus representations and hidden Markov models. In *Wavelets in Medicine and Biology,* A. Aldroubi and M. Unser (Eds.), CRC Press, Boca Raton, FL, 439–464.

© 2008 by Taylor & Francis Group, LLC

W. Sethares. (2007). *Rhythm and Transforms.* Springer, New York, NY.

W. Sethares. (2007). Rhythm and transforms. An extended abstract of his plenary address to the Mathematics and Computation in Music conference in Berlin, May 18, 2007. Available at http://www.mcm2007.info/pdf/fri1-sethares.pdf

E.P. Simoncelli and B.A. Olshausen. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience,* Vol. 24, 1193–1216.

L.M. Smith (2000). A multiresolution time-frequency analysis and interpretation of musical rhythm. Thesis, University of Western Australia.

L.M. Smith's webpage: http://staff.science.uva.nl/~lsmith/

N. Spender. (2004). Music, psychology of. Article in *The Oxford Companion to the Mind,* 625–632, Oxford University Press, Oxford, UK.

G. Strang and T.Q. Nguyen. (1996). *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Boston, MA.

R.S. Strichartz. (1993). How to make wavelets. *The American Math. Monthly,* Vol. 100, 539–556.

W. Sweldens. (1996). The lifting scheme: a custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis,* Vol. 3, 186–200.

W. Sweldens' papers: netlib.bell-labs.com/cm/ms/who/wim/papers/

D.S. Taubman and M.W. Marcellin. (2002). JPEG2000: Image compression fundamentals, standards and practice. Kluwer, Boston, MA.

J. Tian and R.O. Wells, Jr. (1996). A lossy image codec based on index coding. *IEEE Data Compression Conference, DCC '96,* p. 456.

J. Tian and R.O. Wells, Jr. (1998). Embedded image coding using wavelet difference reduction. *Wavelet Image and Video Compression,* P. Topiwala (Ed.), Kluwer, Norwell, MA, 289–301.

L.N. Trefethen. (1998). Maxims about numerical mathematics, computers, science, and life. *SIAM News,* Vol. 31, No. 1.

UCSD video processing page: http://videoprocessing.ucsd.edu/

Michael Unser's webpage: http://bigwww.epfl.ch/unser/

M. Unser and T. Blu. (2003). Wavelet theory demystified, *IEEE Transactions on Signal Processing,* Vol. 51, 470–483.

M. Unser and T. Blu. (2003). Mathematical properties of the JPEG2000 wavelet filters, *IEEE Transactions on Image Processing,* Vol. 12, 1080–1090.

M. Vetterli and J. Kovačević. (1995). *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ.

A. Volk. (2004). Metrical investigations in Brahm's symphonies. In E. Lluis Pueboa, G. Mazzola, and T. Noll, Eds., *Perspectives of Mathematical Music Theory,* epOs Music, Osnabrück.

J.S. Walker. (1988). *Fourier Analysis.* Oxford, New York, NY.

J.S. Walker. (1996). *Fast Fourier Transforms, Second Edition.* CRC Press, Boca Raton, FL.

© 2008 by Taylor & Francis Group, LLC

J.S. Walker. (1997). Fourier analysis and wavelet analysis. *Notices of the Amer. Math. Soc.,* Vol. 44, 658–670.

J.S. Walker. (2000). Lossy image codec based on adaptively scanned wavelet difference reduction. *Optical Engineering,* Vol. 39, 1891–1897.

J.S. Walker. (2002). Tree-adapted wavelet shrinkage. *Advances in Imaging and Electron Physics,* Vol. 104, 343–394.

J.S. Walker and T.Q. Nguyen. (2000). Adaptive scanning methods for wavelet difference reduction in lossy image compression. *IEEE Int'l Conference on Image Processing, Vancouver, Sept. 2000,* Vol. 3, 182–185.

J.S. Walker and T.Q. Nguyen. (2000). Wavelet-based image compression. *Handbook of Transforms and Data Compression,* Chap. 6, 267–312, CRC Press, Boca Raton, FL.

J.S. Walker and Y.-J. Chen. (2006). Denoising Gabor transforms. Preprint. Available at http://www.uwec.edu/walkerjs/media/DGT.pdf

J.S. Walker and G.W. Don. (2006). Music: a time-frequency approach. Preprint. Available at http://www.uwec.edu/walkerjs/Primer/Papers

D.F. Walnut. (2002). *An Introduction to Wavelet Analysis.* Birkhäuser, Boston.

B.A. Wandell. (1995). *Foundations of Vision.* Sinauer Associates, Sunderland, MA.

Z. Wang and A.C. Bovik. (2002). A universal image quality index. *IEEE Signal Processing Letters,* Vol. 9, 81–84.

Z. Wang, A.C. Bovik, and E.P. Simoncelli. (2005). Structural approaches to image quality assessment. Chapter 8.3 in *Handbook of Image and Video Processing, 2nd Edition,* Alan Bovik (Ed.), Academic Press, San Diego, CA.

Z. Wang and E.P. Simoncelli. (2004). Local phase coherence and the perception of blur. *Adv. Neural Information Processing Systems,* Vol. 16.

A.B. Watson. (1987). Efficiency of a model human image code. *J. Optical Soc. Am.,* Vol. 4, 2401–2417.

The *Wavelet Digest* website: http://www.wavelet.org

Why Birds Sing website: http://www.whybirdssing.com/

M.V. Wickerhauser. (1993). Best-adapted wavelet packet bases. In *Different Perspectives on Wavelets,* I. Daubechies (Ed.), AMS, Providence, RI, 155–172.

M.V. Wickerhauser. (1994). *Adapted Wavelet Analysis from Theory to Software.* A.K. Peters, Wellesley, MA.

L. Ying, L. Demanet, and E. J. Candes. (2005). 3D discrete curvelet transform. *Proc. of SPIE—Volume 5914, Wavelets XI,* M. Papadakis, A.F. Laine, and M.A. Unser (Eds.).

R. Young. (1980). *An Introduction to Nonharmonic Fourier Series.* Academic Press, New York, NY.

© 2008 by Taylor & Francis Group, LLC